НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра автоматизованих систем обробки інформації і управління

«На правах рукопису»

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
"___"_____2019 р.

УДК _____

Магістерська дисертація

зі спеціальності *121 «Інженерія програмного забезпечення»*

на тему: «*Математичне та програмне забезпечення для планування задач*»

Виконав: студент VI курсу, групи *ІП-з81мп*

Крикун Євгеній Вікторович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник *доц.,к.т.н. Ліщук К.І .* _____
(посада, науковий ступінь, , вчене звання, прізвище та ініціали) (підпис)

Консультант *доц.,к.т.н. Ліщук К.І* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц.,к.т.н., доц.,к.т.н. Лісовиченко О.І.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти *другий (магістерський) за освітньо-професійною програмою*

Спеціальність *121 «Інженерія програмного забезпечення»*
(код і назва)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
«___» _____ 20*1* р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Крикуну Євгенію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації *Математичне та програмне забезпечення для планування задач*

науковий керівник дисертації *к.т.н., доц. Ліщук К.І.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "____" _____ 20__ р. № _____

2. Строк подання студентом дисертації "____" _____ 20__ р.

3. Об'єкт дослідження *Процес розробки інтелектуальної системи планування та розподілення задач на підприємстві*

4. Предмет дослідження *Методи та інформаційні технології, які використовуються для планування задач при роботі з великою кількістю різнопланового персоналу*

5. Перелік завдань, які потрібно розробити *Аналіз існуючих методів ІПП планування задач на підприємствах, розробка математичного забезпечення для побудови оптимального розподілення робіт між працівниками підприємства, Дослідження ефективності запропонованого методу, розробка обчислювальних алгоритмів та програмного забезпечення*

6. Перелік графічного матеріалу

Схема архітектурного підходу до вирішення задачі

Схема взаємодії компонентів в системі

7. Орієнтовний перелік публікацій *Створення системи ефективного планування задач для робітників шляхом моделювання робочих процесів*, *Системи планування задач та розподілення навантаження між персоналом*

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Графічний			

9. Дата видачі завдання “ *01* ” *вересня* 20 *19* р

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Аналіз існуючих методів проектування системи планування задач*		
2	*Аналіз методів розподілення навантаження*		
3	*Розробка алгоритму розподілення навантаження з великою кількістю додаткових параметрів.*		
4	*Дослідження ефективності розробленого алгоритму.*		
5	*Розробка інформаційного та програмного забезпечення*		
7	*Проведення експериментальних досліджень розробленого алгоритму*		
8	*Розробка технічної документації*		
9	*Подання роботи на попередній захист*	*05.12.2019*	
10	*Подання роботи на основний захист*	*.12.2019*	

Студент _____ (підпис) _____ (ініціали, прізвище)

Науковий керівник _____ (підпис) _____ (ініціали, прізвище)

РЕФЕРАТ

Актуальність теми.

Системи планування задач в сучасному світі займають одну з головних ролей для людини XXI століття. Грамотне розподілення часу на ту чи іншу справу призводить до успішного та правильного способу досягти будь-якої мети. Всі вони зводяться до постановки задачі та часу на її виконання, але майже ніде не враховуються додаткові критерії такі як знання чи навички в тому чи іншому завданні. Наприклад, на підприємстві при розподіленні задач повинна враховуватись не тільки можливість людини виконати поставлену задачу, а дати цю задачу співробітнику з потрібною кваліфікацією, вмінням та бажанням виконувати її.

Таким чином, системи, що передбачають планування та розподілення задач з великою кількістю побічних критеріїв дозволяють підвищити продуктивність та ефективність праці і автоматично стають дуже затребуваними в розробці, а їх впровадження стає актуальною задачею повсякдення.

Мета дослідження: основна мета даної роботи полягає в дослідженні та розробці математичних та програмних засобів для планування та розподілення задач на підприємстві з урахуванням великої кількості додаткових параметрів.

Для досягнення поставленої мети були сформульовані **наступні завдання** дослідження:

— аналіз існуючих методів інформаційної підтримки процесів планування задач на підприємствах;

— розробка математичного забезпечення для побудови оптимального розподілення робіт між працівниками підприємства;

— дослідження ефективності запропонованого методу розподілення задач;

— вибір необхідних програмних засобів для розробки програмного забезпечення розрахункової частини;

— розробка обчислювальних алгоритмів та програмного забезпечення із застосуванням розробленого математичного апарату.

Об'єкт дослідження: процес розробки інтелектуальної системи планування задач на підприємстві.

Предмет дослідження: методи та інформаційні технології, які використовуються для планування задач при роботі з великою кількістю різнопланового персоналу.

Наукова новизна. При проведенні дослідження у дисертаційній роботі використовувались методи планування задач з додатковими параметрами, а саме метод алгоритм багаторівневих черг Multilevel Feedback Queue.

Найбільш суттєвими науковими результатами магістерської дисертації є:

— розроблено математичний метод планування та розподілення задач, який враховує з додаткові параметри, такі як досвід, можливість виконання завдання, тощо;

— модифіковано алгоритм Multilevel Feedback Queue з метою вирішення задачі пошуку найоптимальніших рішень для планування задач.

Практичне значення отриманих результатів визначається тим, що всі запропоновані математичні методи і алгоритми доведені до практичної реалізації у рамках програмного забезпечення, котре використовується для побудови системи планування задач з додатковими параметрами, які впливають на оптимізацію виробничого процесу.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконана в рамках теми «Методи та технології в задачах пошуку та збереження даних. Державний реєстраційний номер 0117U000915».

Апробація. Основні положення роботи доповідались і обговорювались на 3 всеукраїнській науково-практичній конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019)

Публікації. Наукові положення дисертації опубліковані в

Ключові слова.

СИСТЕМИ ПЛАНУВАННЯ ЗАДАЧ, МЕТОД ПЛАНУВАННЯ ЗАДАЧ З ДОДАТКОВИМИ ПАРАМЕТРАМИ, АЛГОРИТМ MULTILEVEL FEEDBACK QUEUE

ABSTRACT

Actuality: Task planning systems in the modern world occupy one of the main roles for the person of the XXI century. A competent allocation of time for a particular cause leads to a successful and correct way to achieve any goal. They all boil down to setting a task and time to complete it, but hardly any additional criteria such as knowledge or skills in a particular task are considered anywhere. For example, in the distribution of tasks, the enterprise should take into account not only the ability of the person to perform the task, but to give this task with the necessary qualifications, skills and desire to perform it.

Thus, systems that allow for the scheduling and allocation of tasks with a large number of side criteria can increase productivity and efficiency and automatically become very demanding in the development, and their implementation is becoming an urgent task everyday.

Purpose of the study: The main purpose of this work is to research and develop mathematical and software tools for the development of a system of planning and distribution of problems with many additional parameters.

To achieve this goal, the following tasks were formulated:

– Investigate existing design programs for task scheduling systems;
– Investigate existing load-sharing algorithms;
– Investigate existing methods of setting systems and task scheduling;
– Develop mathematical support for the task scheduling system;
– Develop software for the task scheduling system;

Research object: Perform an experimental study of the proposed solutions.

Object of study: the process of developing an intelligent task scheduling system in the enterprise

Subject of research: methods and algorithms used to optimize the task scheduling system when working with a large number of diverse staff

Scientific novelty: research and development work in the dissertation used methods of scheduling tasks with additional parameters, namely the method of multilevel queue Multilevel Feedback Queue.

The most significant scientific results of the master's thesis are:

- The method of scheduling methods with additional parameters has been developed.

- Modified the Multilevel Feedback Queue algorithm to solve the problem of finding the best planning solutions.

The practical significance of the obtained results is determined by the fact that all the proposed mathematical methods and algorithms are brought to practical implementation within the software used to build a task scheduling system with additional parameters that influence the optimization of the production process.

Relationship with working with scientific programs, plans, topics:

Testing: The main provisions of the work were reported and discussed at the 3rd All-Ukrainian Scientific and Practical Conference of Young Scientists and Students "Information Systems and Management Technologies" (ISTU-2019)

Publications: Theses of the thesis are published in

Keywords:

TASK PLANNING SYSTEMS, TASK PLANNING METHOD WITH ADDITIONAL PARAMETERS, MULTILEVEL FEEDBACK QUEUE ALGORITHM

Зміст

ВСТУП

Автоматизація планування і раніше була актуальною темою досліджень, оскільки стає необхідним застосування програм-планувальників в нових складних проблемних областях, де постають високі вимоги до якості одержуваних планів. На сучасному рівні розвитку програмування реалізовані в системах штучного інтелекту прийоми і методи планування все ще не адекватні прийомам і методам, застосовуваним людиною при вирішенні складних завдань. Вирішенню цієї проблеми також приділяється велика увага.

Наука про управління підприємством неперервно збагачує арсенал своїх методів і засобів. Причиною появи математичних методів послужило ускладнення економіки та управління господарством. Математичні методи, поєднані із сучасною обчислювальною технікою в рамках різного роду автоматизованих систем, стають надзвичайно важливим елементом фінансового планування та управління на підприємствах, в галузевих та міжгалузевих комплексах. Ці методи все активніше використовуються у практиці та реалізації планів економічного й соціального розвитку. Складовою частиною цього завдання виступає створення єдиної системи оптимального планування задач на базі широкого застосування математичних методів та ЕОМ в повсякденному житті.

Всі сучасні системи планування задач та розподілення навантаження між персоналом мають величезну кількість аналогів і різновидів з реалізації для різних платформ(Android, IOS, Desktop та ін.) які можна використовувати незалежно від місцезнаходження користувача.

Системи планування та розподілення задач існували задовго до появи комп’ютеризованого суспільства та представляли в більшості свої алгоритми для розподілення навантаження по принципу Intelligent Workload Distribution та їх подібних аналогів. Актуальність даної проблеми як для великого так і

для малого та середнього бізнесу являється пріоритетним в розробці систем та підсистем для покрашення та налагодження роботи в компаніях.

Intelligent Workload Distribution(інш. *Intelligent Workload Management*) – це парадигма управління IT-системами, що виникає в результаті перетину динамічної інфраструктури, віртуалізації, управління ідентифікацією та дисципліни розробки програмного забезпечення. IWM забезпечує керування та оптимізацію обчислювальних ресурсів у безпечному та сумісному режимі у фізичних, віртуальних та хмарних середовищах для надання бізнес-послуг кінцевим клієнтам. Парадигма IWM спирається на традиційну концепцію управління робочим навантаженням, за допомогою якої ресурси обробки обробляються динамічно, або "робочі навантаження", засновані на таких критеріях, як пріоритети бізнес-процесів (наприклад, у балансуванні запитів бізнес-аналітики на обробку онлайн-транзакцій) наявність ресурсів, протоколи безпеки або планування подій, але розширює поняття в структуру окремих робочих навантажень.

Сучасні системи планування задач при їх розподіленні в основному опираються на рішення конкурентів та подібні, які були розроблені до них. Тому не існує сучасного рішення для розподілення та планування задач, яке враховувало б всі аспекти задачі. Тому з впевненістю можна сказати, що така система була б вкрай необхідною для використання у всіх сферах життя починаючи від особистого планування, закінчуючи плануванням та розподіленням задач на великому підприємстві.

Таким чином, системи, що передбачають планування та розподілення задач з великою кількістю побічних критеріїв дозволяють підвищити продуктивність та ефективність праці і автоматично стають дуже затребуваними в розробці, а їх впровадження стає актуальною задачею повсякдення.

Тому основна мета даної роботи полягає в дослідженні та розробці математичних та програмних засобів для розробки системи планування та розподілення задач з великою кількістю додаткових параметрів.

Для досягнення поставленої мети необхідно розв‘язати низку наступних задач:

— дослідити існуючі програми проектування системи планування задач;

— дослідити існуючі алгоритми розподілення навантаження;

— дослідити існуючі методи побудови систем постановок та планування завдань;

— розробити математичне забезпечення для системи планування задач;

— розробити програмне забезпечення для системи планування задач;

— виконати експериментальне дослідження запропонованих рішень.

1 ОГЛЯД ПІДХОДІВ ДО СУЧАСНОГО БАЧЕННЯ СИСТЕМ ПЛАНУВАННЯ ЗАДАЧ

1.1 Загальні положення

Сучасні системи планування задач використовують багато різних методів та варіацій розробок та реалізацій цієї проблеми. В нинішній час вони можуть бути доступними як на комп’ютері, так і на мобільних пристроях і навіть на смарт – годинниках.

Ефективне планування завдань вимагає управління всіма аспектами задачі, включаючи його стан, пріоритет, час, призначення людських та фінансових ресурсів, періодичність, залежність, повідомлення тощо. Вони можуть бути об'єднані широко в основні напрямки управління завданнями.

Управління декількома особами або завданнями команди можуть допомогти спеціалізованим програмним забезпеченням, наприклад, робочим процесом або програмним забезпеченням для управління проектами. Насправді багато людей вважають, що управління завданнями повинно слугувати фундаментом для діяльності з управління проектами.

Управління завданнями може бути частиною управління проектами та управління процесами і може слугувати основою для ефективного робочого процесу в організації. Керівники проектів, які дотримуються управління, орієнтованого на завдання, мають детальний та сучасний графік проекту, і, як правило, добре спрямовують членів команди та рухають проект вперед.

1.2 Огляд існуючих рішень систем планування навантаження

1.2.1 Десктопні рішення систем планування навантаження

Системи розподілення та планування навантаження на даний момент представлені в різноманітному вигляді, проте декілька з них виділяються унікальністю представлення і їх можна розділити на 2 підвиди:

— системи, які базуються на розподіленні часу;

— системи, які базуються на розподіленні задач.

Наприклад, система планування Saviom, 10,000ft, Resource Guru та Forecast мають дуже схожий за зовнішнім виглядом користувацький інтерфейс, проте різняться функціоналом, який пропонують розробники цих продуктів.

Наприклад, система Saviom (Рисунок 1.1) має в своєму арсеналі інтелектуальні функції для розподілення не тільки часу на роботу працівників, але і детальні огляди відпрацьованого часу, та розподілення задач відповідно до кваліфікацій працівників. В системі доступна така можливість як пріорітеторізація задач для обраних проектів та можливість одночасно займатися та розподіляти завдання на менеджмент декількох проектів. Ця програма являється необхідним додатком для будь-якої компанії зі штатом понад 100 робітників.

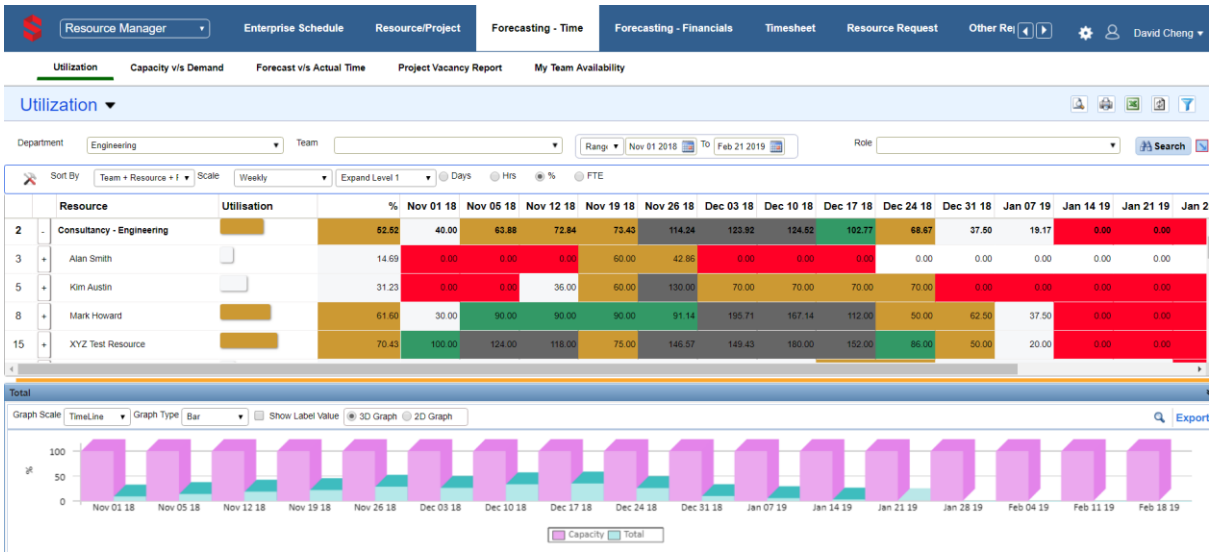


Рисунок 1.1 – Система Saviom

Система 10,000ft (Рисунок 1.2) має дещо простіший функціонал, ніж Saviom, проте це не заважає виконувати поставлені для неї задачі. Ця система пропонує легший в настроюванні та використанні функціонал та дещо легший в освоєнні та наглядності користувацький інтерфейс. 10,000ft має унікальну функцію порівняння проектів і ресурсів, яка допомагає підібрати окремих людей до вимог проекту, знаходячи членів команди відповідно до різних критеріїв, таких як дисципліни, навички, доступність та інші спеціальні критерії.

Завдяки вбудованим розкладам, мобільному відстеженню часу та відстеженню витрат можна створювати звіти про багаті проекти, фільтруючи дані проекту лише за кілька кліків. Звіти можуть дати вам огляд задач або прогнозований вигляд таких речей, як використання команди, фактичні або планові звіти про час, відстеження бюджету, звіти про витрати та проекти в процесі підготовки.

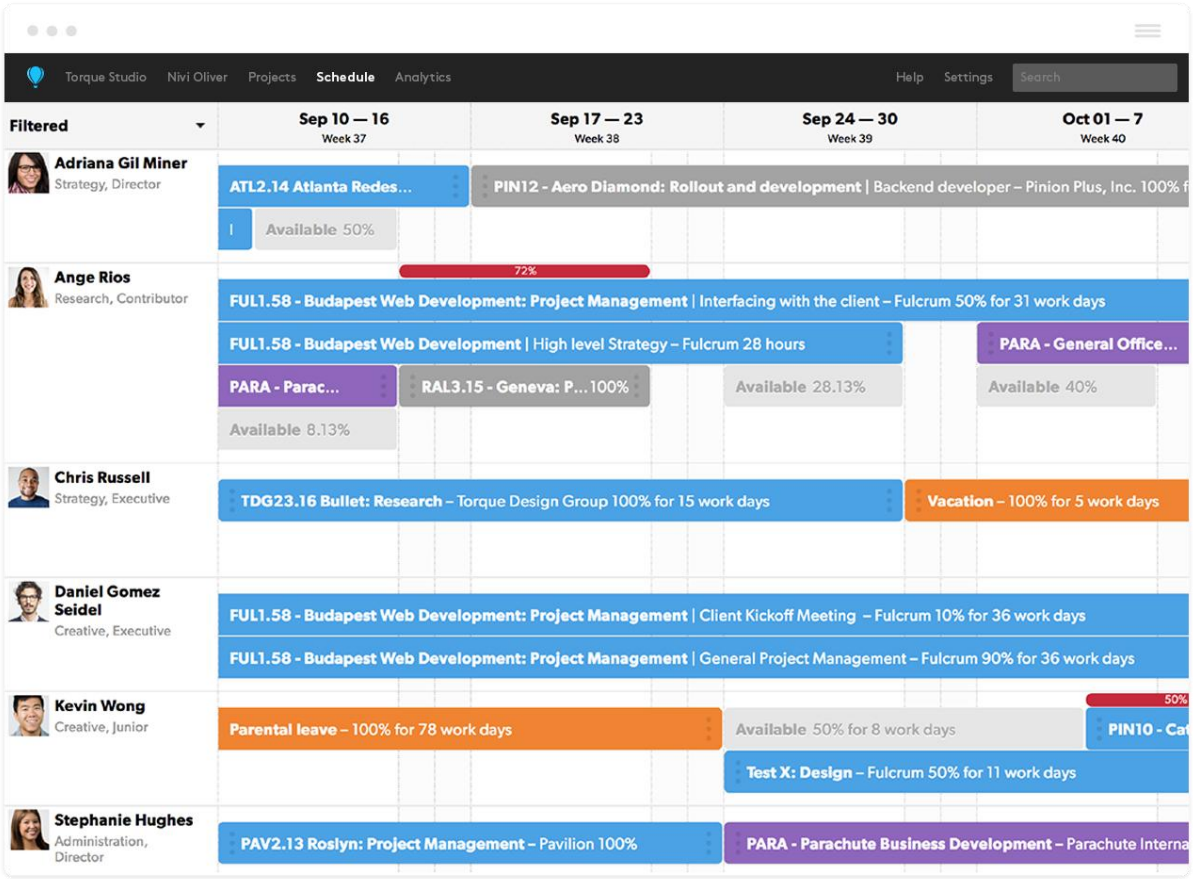


Рисунок 1.2 – Система 10,000ft

Система Resource Guru (Рисунок 1.3) пропонує деякі ґрунтовні функції, включаючи: візуальний інтерфейс стилю календаря, користувацькі перегляди, поля та фільтри, індикатор доступності, тощо. Resource Guru має сенс продемонструвати, що є вирішенням конфлікту ресурсів: управління ресурсами та колективне співробітництво побудовано для того, щоб зробити резервування ресурсів більш простим, з управлінням конфліктами, управлінням відпустками та списком очікування для запобігання надмірного бронювання. Вона допомагає менеджерам проектів робити замовлення

одночасно, не маючи жодних шансів наступити один одному на ноги. Зіткнення автоматично запобігаються.

Доречно, кожен співробітник отримує власну інформаційну панель ресурсів, щоб вони могли входити в систему і точно знати, над чим вони повинні працювати. А для моніторингу ефективності бізнесу потужні звіти допомагають визначити, які проекти та клієнти перенасичуються, а також допомагають контролювати використання команди та допомагають у майбутньому планувати потужності.

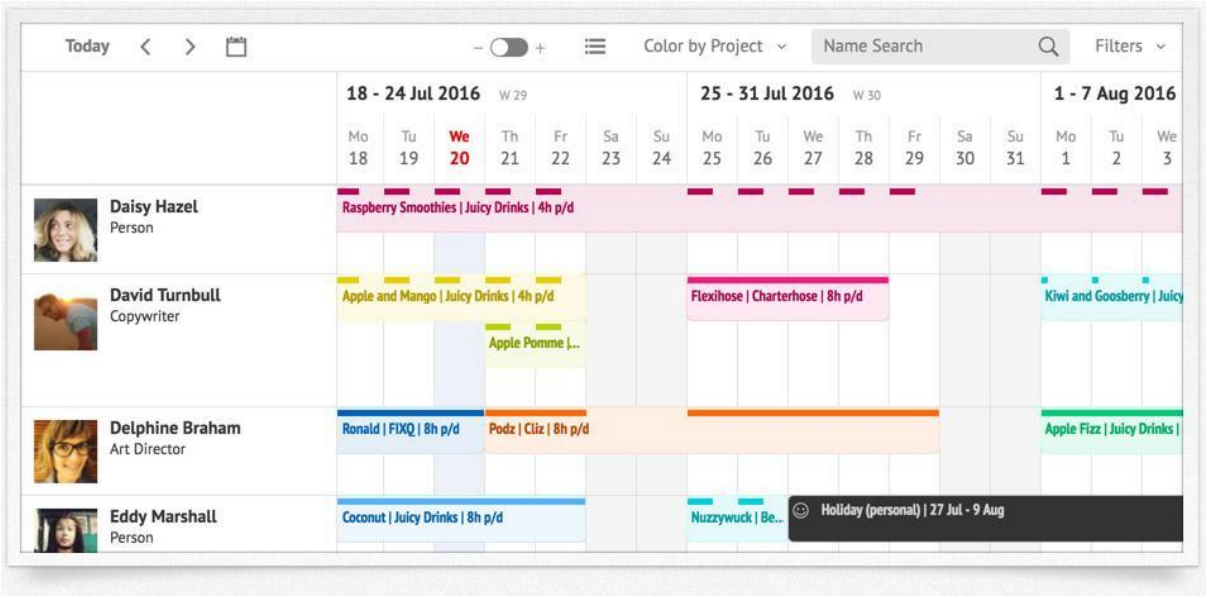


Рисунок 1.3 – Система Resource Guru

Система Forecast це повне рішення для управління ресурсами та проектами, яке використовують агенції та професійні сервісні компанії по всьому світу, і це популярне програмне забезпечення для управління IT-ресурсами. Для ефективного управління ресурсами і планування людей ви можете замовити людей до проектів і ви можете перетасувати завдання в декількох проектах, щоб переконатися, що ніхто не має занадто мало або занадто багато роботи в певний час. Все це здійснюється за допомогою інтерфейсу перетягування, що робить його інтуїтивним у використанні. Швидке бронювання дійсно корисне - це допоможе вам швидко знайти людину з правильною роллю або навичками, які, швидше за все, будуть доступні для даного завдання. Розмітки та відстеження вбудовані

безпосередньо в продукт, а програми для iOS і Android можна використовувати для планування та роботи на ходу. Ви можете створювати та фільтрувати власні ідеї та безпечно ділитися з клієнтами та зацікавленими сторонами. Вони дають змогу побачити такі речі, як використання в реальному часі, порівняти фактичне або заплановане використання, відстежувати прогрес проекту, прибутковість, бюджети тощо. Глибокі інтеграції роблять Forecast ідеальним інструментом планування ресурсів для JIRA, а також іншими продуктами, переліченими в розширеному каталозі програм Forecast.

1.2.2 Огляд існуючих систем планування навантаження для мобільних пристроїв.

Мобільні пристрої в сучасному світі досягли того рівня, на якому вони можуть бути суперниками звичайних комп’ютерів в простих рішеннях на рівні зручного редагування тексту чи менеджменту задач. В сумі з великою кількістю засобів комунікування на самому пристрої, а також величезною мобільності. Можна впевнено сказати, що дані девайси можуть бути конкурентними і в системі планування менеджменту. Проте ринок програм для налагодження ефективної роботи команди за допомогою мобільного пристрою не такий широкий. Розглянемо декілька рішень мобільного менеджменту на прикладі додатків для смартфонів:

— Wrike;

— Trello;

— Asana.

Всі вказані вище додатки реалізують клієнт – серверну модель взаємодії між користувачами, а також являються мобільними копіями десктопних рішень, які доповнюють та частково заміняють повноцінні додатки при відсутності доступу до них. Також потрібно виділити декілька особливостей використання даних мобільних систем:

— потрібно постійне підключення до мережі інтернет;

— потрібен сервер для обробки та зберігання даних по пріоретизації та розподіленні навантажень;

— використання мобільної системи може повністю не надавати доступу до всього функціоналу, як в десктопній версії.

Прикладом для огляду може слугувати система Wrike (Рисунок 1.4, 1.5). Незалежно від того, чи всі ваші співробітники працюють в офісі або у вас є працівники, які працюють віддалено, Wrike - це чудовий інструмент управління проектами для оптимізації вашого робочого процесу. Завдяки інструментам для керування завданнями, співпраці з документами, моніторингу робочого навантаження та спеціальних звітів, це дозволить вам планувати, розставляти пріоритети та відстежувати ваші проекти на кожному кроці. Вона навіть має в реальному часі стрічку новин і інтерактивну шкалу часу, щоб дозволити гнучке управління проектами.

Якщо у вас немає деяких функцій, які ви шукаєте, ви можете використовувати його відкриті можливості API, щоб об'єднати платформу з деякими з ваших улюблених інструментів підвищення продуктивності. Таким чином, ви можете мати більш персоніфікований користувальницький досвід, який відповідає вимогам ваших операцій.

Wrike пропонує безкоштовний пакет, який ідеально підходить для невеликих команд до п'яти користувачів. Для великих компаній, ви можете вибрати будь-який з їх платних планів, які оцінюються відповідно до типів функцій які вам потрібні, кількість користувачів, які потрібно зареєструвати в системі, і простір для зберігання що вам потрібен.

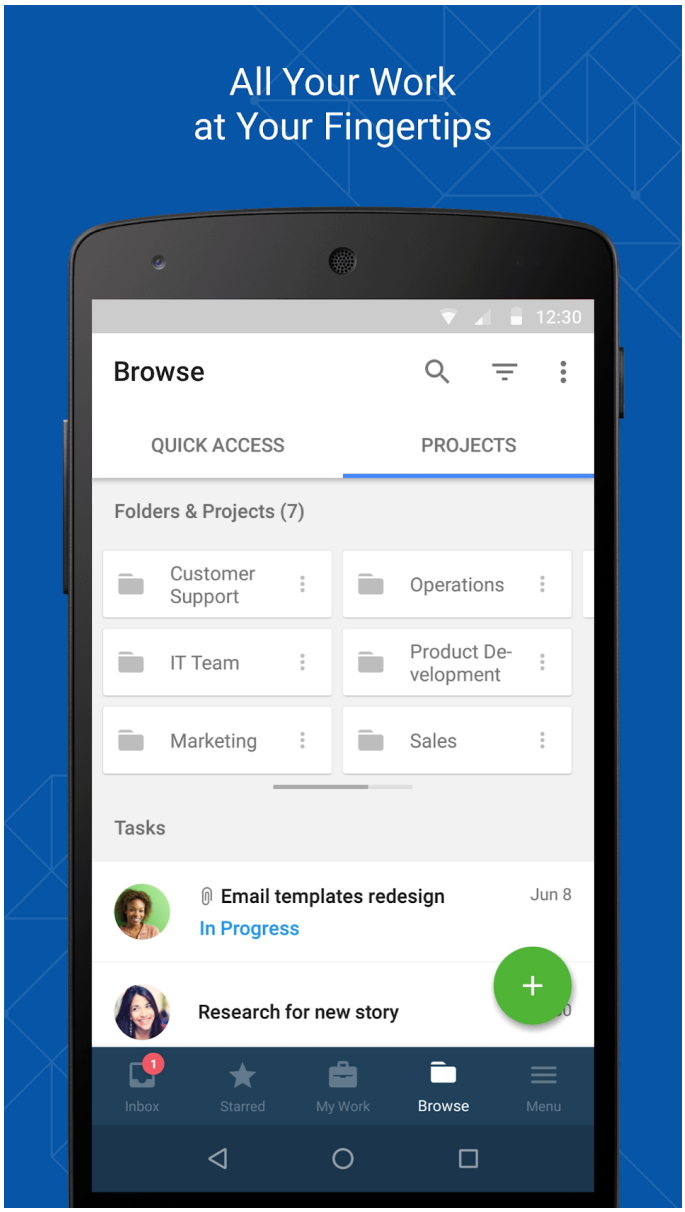


Рисунок 1.4 – Система Wrike

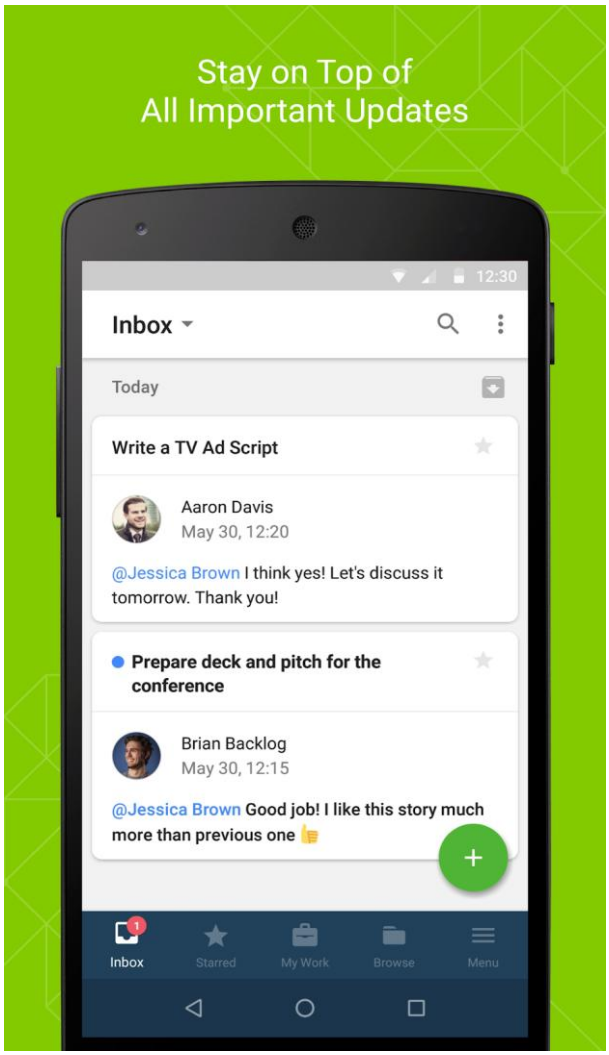


Рисунок 1.5 – Система Wrike

Система Trello (Рисунок 1.6) – це провідна система управління проектами та система співпраці, ця система пропонує зручну систему карт з функціональністю, що дозволяє відмовитися від редагування. Він працює як передовий список завдань з інструментами для призначення завдань, журналу діяльності, сповіщення електронною поштою та спільної роботи. Пропонуючи простий у використанні інтерфейс, Trello залишається дуже гнучким додатком через кількість функцій, які він може підтримувати. Незалежно від того, чи потрібні Google Drive, Evernote, Github або Salesforce, ця платформа може інтегруватися з потрібними інструментами, щоб додатково спростити робочий процес. Крім того, вся система захищена шифруванням SSL, щоб зберегти дані вашої компанії в будь-який час.

Trello пропонує назавжди безкоштовний план, який може вмістити стільки дощок, карт і команд, скільки вам потрібно. Це включає файлові вкладення розміром до 10 МБ, а також основні функції, які може запропонувати платформа.

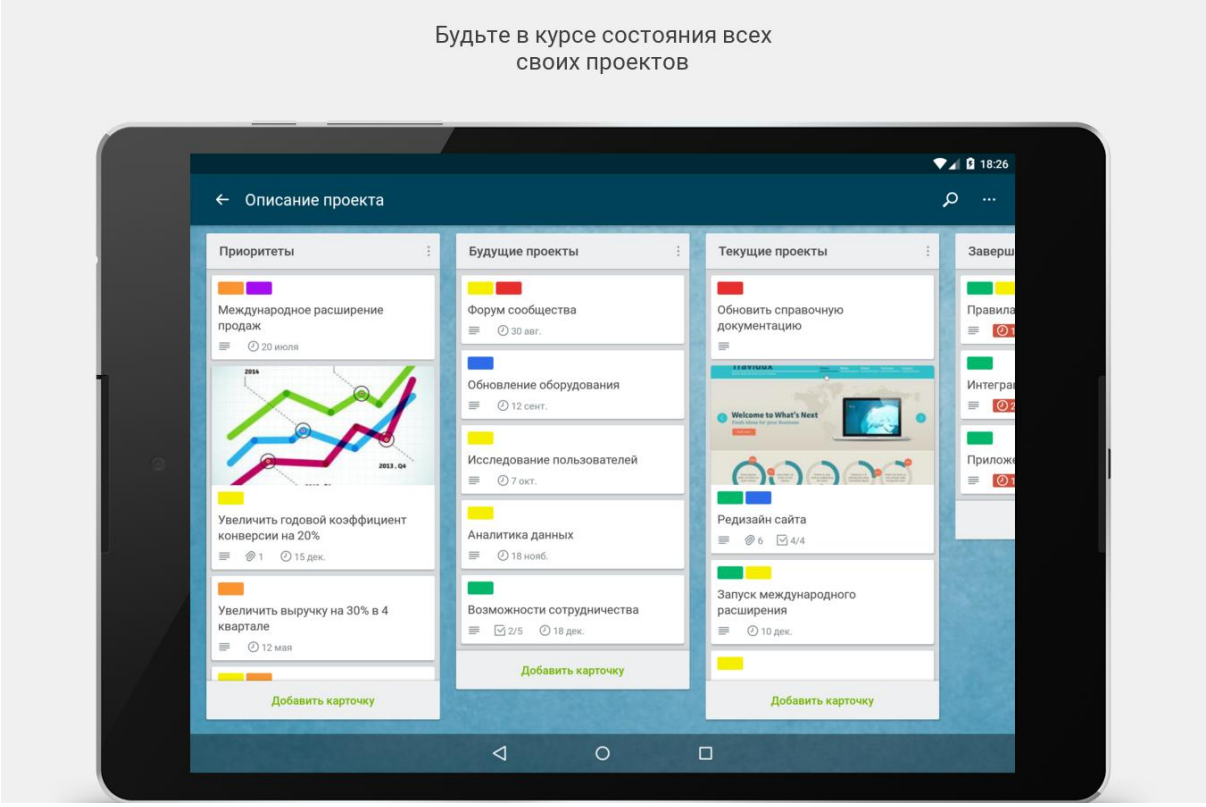


Рисунок 1.6 – Система Trello

Система Asana (Рисунок 1.7, 1.8) – це популярна програма управління проектами, ця платформа зосереджена в основному на управлінні завданнями. Він пропонує численні робочі області, створення діаграми Ганта, управління таблицями Канбан і оновлення в реальному часі, які допоможуть вам більш ефективно контролювати ваші проекти. Він навіть має функцію режиму фокусування, яка дозволяє вам і вашим товаришам по команді обнуляти пріоритетні завдання. Доступ до всіх цих функцій можна отримати за допомогою власного програмного забезпечення для Android.

Платформа також побудована для усунення необхідності інтеграції електронної пошти та програм сторонніх розробників для комунікації. Вона вже оснащена функціями коментарів, функціями розмови та іншими інструментами співпраці, які допоможуть вам працювати з партнерами. Якщо

ви вважаєте, що вбудованих засобів комунікації цієї системи недостатньо, платформа все ще є достатньо потужною для інтеграції з різними системами, такими як Google Drive, Dropbox, Slack, Zapier та інші.

Для ціноутворення, Asana пропонує безкоштовний план для команд до 15 членів. Це може підтримувати основні панелі інструментів і необмежену кількість завдань, проектів і розмов.

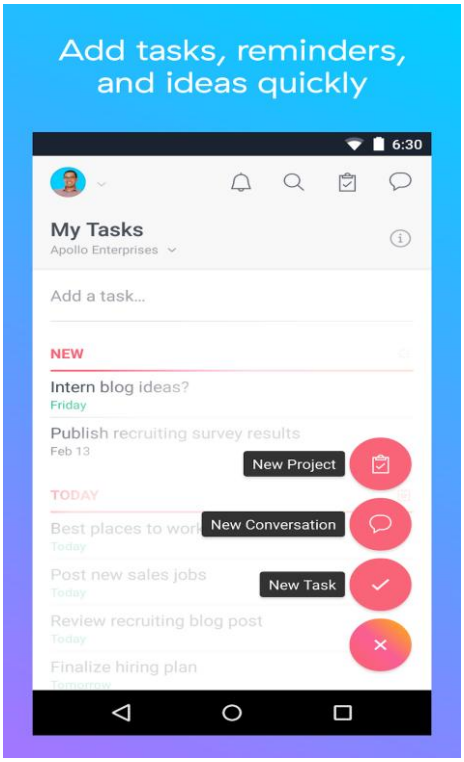


Рисунок 1.7 – Система Asana

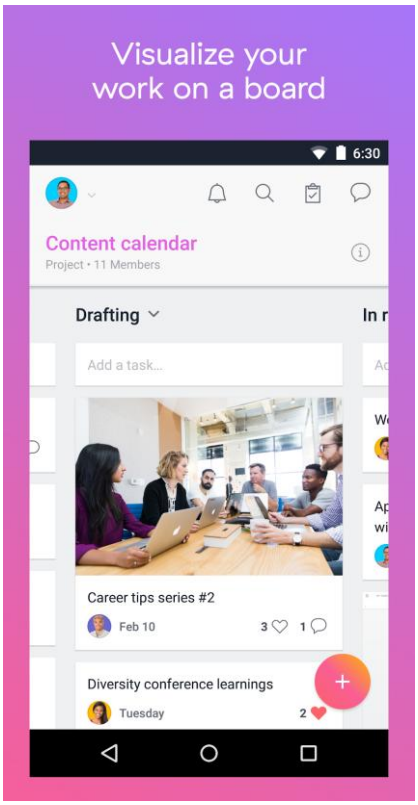


Рисунок 1.8 – Система Asana

1.3 Огляд існуючих алгоритмів систем планування задач

Враховуючи специфіку даної теми, а також розглянувши існуючі рішення можемо сказати, що системи такого типу не використовують один чи декілька готових алгоритмів. Деякі системи використовують штучний інтелект для розподілення навантажень, в той час як інші – в напівавтоматичному режимі за допомогою людини прораховують всі можливі варіанти розподілення. Проте є декілька варіантів розподілення навантажень в залежності від критеріїв вхідних даних, які можуть бути використані для покращення або ж розробки програмного забезпечення, яке не має аналогів.

1.3.1 Алгоритм FCFS

FCFS (first come – first serve; першим прийшов – першим обслуговується) – найпростіший алгоритм, робота якого зрозуміла з назви. Це алгоритм без витіснення, тобто процес, обраний для виконання на Центральному процесорі(далі ЦП), не переривається, поки не завершиться (чи не перейде в стан очікування по власної ініціативи). FCFS забезпечує мінімум накладних витрат. Середній втрачений час цього алгоритму не залежить від тривалості процесу, але штрафне відношення при рівному втраченому часі буде великим для коротких процесів. Тому алгоритм FCFS вважається найкращим для довготривалих процесів. Істотною перевагою цього алгоритму наряду з його простотою являється та обставина, що FCFS гарантує відсутність нескінченного відкладання процесів: інший зайшовший в систему процес буде оброблений, в кінці кінців, незалежно від степені завантаження системи.

1.3.2 Алгоритм RR

RR (round robin - карусель) - найпростіший алгоритм з витісненням. Процес отримує в своє розпорядження ЦП на деякий квант часу Q (в найпростішому випадку розмір кванта фіксований). Якщо за час Q процес не завершився, він витісняється з ЦП і направляється в кінець черги готових

процесів, де чекає виділення йому наступного кванта, і т.д. Показники ефективності RR істотно залежать від вибору величини кванта Q. RR забезпечує найкращі показники, якщо тривалість більшості процесів наближається до розміру кванта, але не перевершує його. Тоді більшість процесів укладаються в один квант і не стають в чергу повторно. При величині кванта, яка прагне до нескінченності, RR вироджується в FCFS. При Q, яка прагне до 0, накладні витрати на перемикання процесів зростають настільки, що поглинають весь ресурс ЦП. RR забезпечує найкращі показники справедливості: штрафне відношення P на великій ділянці тривалостей процесів t залишається практично постійним. Тільки на ділянці t <Q штрафне відношення починає змінюватися і при зменшенні t від Q до 0 зростає експоненціально. Втрачений же час M істотно зростає зі збільшенням тривалості процесу.

1.3.3 Алгоритм SJF

SJF (shortest job first - найкоротша робота - першої) - невитісняючий алгоритм, в якому найвищий пріоритет має найкоротший процес. Для того щоб застосовувати цей алгоритм, повинна бути відома тривалість процесу – задаватися користувачем або обчислюватися методом екстраполяції. Для коротких процесів SJN забезпечує найкращі показники, ніж RR, як по втраченому часу, так і по штрафному відношенню. SJN забезпечує максимальну пропускну здатність системи - виконання максимального числа процесів в одиницю часу, але показники для довгих процесів значно гірші, а при високому ступені завантаження системи активізація довгих процесів може відкладатися до нескінченності. Штрафне ставлення слабо змінюється на основному інтервалі значень t, але значно зростає для найкоротших процесів: такий процес при вступі до системи має найвищий пріоритет, але змушений чекати, поки закінчиться поточний активний процес.

1.3.4 Алгоритм PSJF

PSJF (preemptive SJN - SJN з витісненням) - поточний активний процес переривається, якщо його час, що залишив на виконання більше, ніж у новоприбулого процесу.

Алгоритм забезпечує ще більшу перевагу коротким процесам перед довгими. Зокрема, в ній усувається те зростання штрафного відношення для найкоротших процесів, яке має місце в SJN.

1.3.5 Алгоритм RR SJF

Модифікація алгоритму RR з переупорядковуванням процесів в черзі відповідно до часу, що залишився на виконання.

1.3.6 Алгоритм HPRN

HPRN (highest penalty ratio next - з найбільшим штрафним ставленням - наступний) - алгоритм без витіснення, що забезпечує найкращі показники справедливості. Це досягається за рахунок динамічного перевизначення пріоритетів. Всякий раз при звільненні ЦП для всіх готових процесів обчислюється поточне штрафне відношення

$$p[i] = (w[i] + t[i]) / t[i],$$

де i - номер процесу; w [i] - час, витрачений процесом на очікування; t [i] - тривалість процесу прогнозована. Для щойно надійшовшого процесу p [i] = 1. ЦП віддається процесу, що має найбільше значення p [i]. Для коротких процесів HPRN забезпечує приблизно ті ж показники справедливості, що і SJN, для довгих - ближчі до FCFS. На великому діапазоні середніх тривалостей процесів показники, що забезпечуються HPRN, представляють середнє між SJN і FCFS і слабо залежать від тривалості. Ще одна перевага HPRN в тому, що в часі очікування може враховуватися (з деякими ваговими коефіцієнтами) і очікування в інших чергах і, таким чином, виконується більш повний облік завантаження системи.

Істотним недоліком методу є необхідність переобчислення штрафного відношення для всіх процесів при кожному перемиканні, що погано

узгоджується із загальною політикою мінімізації накладних витрат в дисциплінах без витіснення.

1.3.7 Алгоритм SRR

SRR (selfish RR - егоїстичний RR) - метод з витісненням, що дає додаткові переваги виконуваних процесів, що дозволяє підвищити пропускну здатність. Всі процеси поділяються на дві категорії: нові і вибрані. Новими вважаються ті процеси, які не отримали ще жодного кванта часу ЦП, все інші процеси - вибрані. При надходженні в систему кожному процесу дається деякий пріоритет P0, однаковий для всіх процесів, який в подальшому зростає. В кінці кожного кванта часу перераховуються пріоритети всіх процесів, причому пріоритети нових процесів зростають на величину dA, а обраних - на величину dB. ЦП віддається процесу з найвищим пріоритетом, а за однакової кількості пріоритетів - тому, який раніше поставлений в чергу. Показники алгоритму істотно залежать від обраного співвідношення між dA і dB. При dB / dA = 0 алгоритм вироджується в звичайний RR, при dB / dA> = 1 - в FCFS. Власне алгоритм SRR забезпечується в діапазоні значень 0 <dB / dA <1.

1.3.8 Лотерейне планування

Процесам роздаються "лотерейні квитки" на доступ до ресурсів. Планувальник може вибрати будь-який квиток, випадковим чином. Чим більше квитків у процесу, тим більше у нього пріоритет. Розподіл квантів часу зазвичай здійснюють за алгоритмом RR з пріоритетами.

1.3.9 Алгоритм HLRR

HLRR ("half-life round-robin"). Алгоритм напіврозпаду є модифікацією алгоритму RR. З кожним i-м процесом пов'язано деякий пріоритетне число P [i]. Чим воно менше, тим вище пріоритет процесу. Кожен новий процес отримує деяке початкове значення пріоритетного числа P0, однакове для всіх процесів. Крім того, з кожним процесом пов'язаний лічильник процесорного

часу U [i] з вихідним значенням 0. Процес з найменшим значенням P [i] отримує квант часу Q (за однакової кількості пріоритетних чисел ЦП віддається процесу, який найдовший). За час кванта інтервальний таймер видає кілька сигналів-переривань з інтервалом dT. За кожним таким переривання лічильник U [i] активного (тільки активного!) Процесу збільшується на 1.

Використання ЦП процесом закінчується при закінченні кванта або при переході процесу в очікування. При цьому модифікуються лічильники процесорного часу всіх (в тому числі і неактивних) процесів:

$$U [i] = U [i] / 2$$

і для всіх процесів переобчислюють пріоритетні числа:

$$P [i] = P0 + U [i] / 2$$

і модифікується черга виконання процесів.

1.3.10 Багаторівневі черги

Для систем, в яких процеси можуть бути легко розсортовані на різні групи, був розроблений інший клас алгоритмів планування. Для кожної групи процесів створюється своя чергу процесів, що знаходяться в стані готовності. Цим чергам приписуються фіксовані пріоритети. Наприклад, пріоритет черги системних процесів встановлюється більше, ніж пріоритет черг призначених для користувача процесів. А пріоритет черги процесів, запущених студентами, - нижче, ніж для черги процесів, запущених викладачами. Це означає, що жоден призначений для користувача процес не буде обраний для виконання, поки є хоч один готовий системний процес, і жоден студентський процес не отримає в своє розпорядження процесор, якщо є процеси викладачів, готові до виконання. Усередині цих черг для планування можуть застосовуватися найрізноманітніші алгоритми. Так, наприклад, для великих рахункових процесів, які не потребують взаємодії з користувачем (фонових процесів), може використовуватися алгоритм FCFS, а для інтерактивних процесів - алгоритм RR. Подібний підхід, що отримав

назву багаторівневих черг, підвищує гнучкість планування: для процесів з різними характеристиками застосовується найбільш підходящий їм алгоритм.

1.3.11 Алгоритм FB

FB (foreground-background - передній-задній плани) - черга готових процесів розщеплюється на дві підчерги - черга переднього плану і черга заднього плану.

Черги обслуговуються по алгоритмам RR, але чергу переднього плану має абсолютний пріоритет: поки в ній є процеси, чергу заднього плану не обслуговується. Новий процес прямує в чергу переднього плану. Якщо процес використовував встановлене число N квантів в черзі переднього плану, але не завершився, він переводиться в чергу заднього плану.

1.3.12 Багаторівневі черги зі зворотним зв'язком (Multilevel Feedback Queue)

Подальшим розвитком алгоритму багаторівневих черг є додавання до нього механізму зворотного зв'язку. Тут процес не постійно приписаний до певної черги, а може мігрувати з черги в чергу, в залежності від своєї поведінки.

Для простоти розглянемо ситуацію, коли процеси в стані готовність організовані в 4 черги. Планування процесів між чергами здійснюється на основі витісняючого пріоритетного механізму. Чим вище розташовується черга, тим вище її пріоритет. Процеси в черзі 1 не можуть виконуватися, якщо в черзі 0 є хоча б один процес. Процеси в черзі 2 цієї статі не будуть обрані для виконання, поки є хоч один процес в чергах 0 і 1. І, нарешті, процес в черзі 3 може отримати процесор в своє розпорядження тільки тоді, коли черги 0, 1 і 2 порожні. Якщо при роботі процесу з'являється інший процес з будь-якої більш пріоритетної черги, що виконується процес витісняється. Планування процесів всередині черг 0-2 здійснюється з використанням алгоритму RR, планування процесів в черзі 3 ґрунтується на алгоритмі FCFS.

Створений процес надходить в чергу 0. При виборі на виконання він отримує в своє розпорядження квант часу розміром Q одиниць. якщо тривалість його CPU burst менше цього кванта часу, процес залишається в черзі 0. В іншому випадку, він переходить в чергу 1. Для процесів з черги 1 квант часу має величину 2Q. Якщо процес не вкладається в цей час, він переходить в чергу 2. Якщо укладається - залишається в черзі 1. У черзі 2 величина кванта часу становить 4Q одиниці. Якщо і цього мало для безперервної роботи процесу, процес надходить в чергу 3, для якої квантування часу не застосовується, і, за відсутності готових процесів в інших чергах, він може виконуватися до закінчення свого CPU burst. Чим більше значення тривалості CPU burst, тим в менш пріоритетну чергу потрапляє процес, але тим на більшу процесорний час він може розраховувати для свого виконання. Таким чином, через деякий час все процеси, що вимагають малого часу роботи процесора виявляться розміщеними в високопріоритетних чергах, а всі процеси, що вимагають великого рахунку і з низькими запитами до часу відгуку - в фонових. Організація переміщення процесів з черг з низькими пріоритетами в черги з великими пріоритетами дозволяє більш повно враховувати зміну поведінки процесів з плином часу.

1.4 Висновки до розділу

Отже, враховуючи всі вищерозглянуті системи, можна з впевненістю сказати, що більшість з них мають багато спільних недоліків, а саме:

— функціонал для вирішення завдань по плануванню навантаження побудований на принципі не інтелектуального вирішення завдань, а розподілення в напівавтоматичному або ручному режимі;

— системи мають величезну кількість функціоналу, який інколи загромаджує сприйняття і не дозволяє повністю сфокусуватися на вирішенні проблеми;

— мобільні системи залежні від існування серверного додатку і підтримують не весь функціонал повноцінних десктопних рішень;

— розподілення задач в низці продуктів не підтримує вибору кваліфікаційного рівню працівників;

З плюсів таких систем можна виділити наступне:

— системи даного типу(як мобільні так і десктопні) мають великий потенціал в плані значного полегшення розробки і упорядкування проектів;

— планування навантаження скорочується в часі після введення однієї чи декількох систем на виробництво;

— декілька з приведених систем мають високу мобільність для вирішення поставлених задач і можуть бути використані без наявності комп’ютера.

Проаналізувавши алгоритми для отримання готового рішення можна зробити висновок що поки що не існує готового рішення для вирішення даної задачі.

Тому необхідно розробити своє рішення на основі наведених додатків та кардинально допрацювати готові алгоритми для розподілення навантажень. За основу було прийнято взяти алгоритм Multilevel Feedback Queue.

2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗОВАНОГО ПЛАНУВАННЯ ЗАДАЧ

На сьогодні існує величезна кількість варіантів побудови систем планування задач, котрі займаються розподіленням та упорядкуванням поставлених завдань. Такі системи бувають для розподілення задач вручну між робітниками так і за допомогою інтелектуальних засобів. Нас цікавлять інтелектуальні системи, а саме системи для розподілення задач по пріорітетах з додатковими параметрами. При її вирішенні шукають оптимальне призначення з умови максимуму загальної продуктивності, яка дорівнює сумі продуктивності виконавців. Найбільш ефективним методом її вирішення є Multilevel Feedback Queue метод. Питання про планування задач має багато інтерпретацій: розподіл робіт між механізмами, розподіл цілей між вогневими засобами для максимізації математичного очікування числа уражених цілей або середнього збитку і т.д.

У найбільш загальній формі завдання формулюється в такий спосіб: є деяке число робіт і деяке число виконавців. Будь-який виконавець може бути призначений на виконання будь-якої (але тільки однієї) роботи, але з неоднаковим рівнем складності. Потрібно розподілити роботи так, щоб виконати роботи з мінімальними витратами. Якщо число робіт і виконавців збігається, то завдання називається лінійним завданням про призначення. Зазвичай, якщо говорять про завдання про призначення без додаткових умов, мають на увазі лінійну задачу про призначення.

2.1 Алгоритм Multilevel Feedback Queue

Multilevel Feedback Queue - алгоритм оптимізації, що вирішує завдання про призначення за поліноміальний час. Він був розроблений і опублікований Харольдом Кіном в 1955 році.

В 1967 році вчені припустили, що алгоритм є строго поліноміальним (який згодом таким і виявився). З цього часу алгоритм відомий також як алгоритм Кіна для рішення задачі про розподілення. Тимчасова складність оригінального алгоритму була O (n^4), проте Едвард Скарпа (а також Джон Мнемоні незалежно від них) показали, що його можна модифікувати так, щоб досягти часу виконання O (n^3). Геральд Файт і Едвард Сіккер поширили метод на загальні транспортні завдання. У 2001 році було виявлено, що Карл Фрідріх Гаус знайшов рішення задачі про призначення в XIX столітті і опублікував його в 1790 році на латині.

Алгоритм простіше описати, якщо сформулювати завдання, використовуючи дводольний граф. Дан повний двочастковий граф G = (S, T; E) з n вершинами, відповідними працівникам (S), і n вершинами, відповідними видами робіт (T); вартість кожного ребра c (i, j) невід'ємна. Потрібно знайти досконале, або повне збігання пар з найменшою вартістю.

Будемо називати функцію y: $(S \cup T) \to R$ потенціалом, якщо

$$y(i) + y(j) \leq c(i,j)$$

для кожного $i \in S, j' \in T$. Значення потенціалу(1) дорівнює

$$\sum_{\vartheta \epsilon S \cup T} \quad y(\vartheta). \quad (1)$$

Неважко помітити, що вартість будь-якого досконалого збігання пар не менше, ніж значення будь-якого потенціалу. Угорський метод знаходить повне збігання пар і потенціалів з однаковою вартістю / значенням, що доводить оптимальність обох величин. Фактично він знаходить зроблене парополучення жорстких ребер: ребро ij називається жорстким для потенціалу y, якщо $y(i) + y(j) = c(i,j)$. Підграф жорстких ребер будемо позначати як G_y. Вартість повного парополучення в G_y (якщо воно існує) дорівнює значенню y.

2.2 Алгоритм в термінах дводольних графів

Під час роботи алгоритму ми підтримуємо потенціал y та орієнтацію G_y (позначається $\overrightarrow{G_y}$), який має властивість, що краї, орієнтовані від T до S, утворюють відповідність M. Спочатку y скрізь 0, а всі ребра орієнтовані від S до T (тому M порожній). На кожному кроці ми або модифікуємо y так, щоб його значення збільшувалося, або змінюємо орієнтацію для отримання відповідності з більшою кількістю ребер. Ми вважаємо інваріантним, що всі ребра M є жорсткими. Ми закінчуємо, якщо M - ідеальне паросполучення.

В загальному нехай $R_S \subseteq S$ та $R_T \subseteq T$- вершини, не охоплені M (так R_S складається з вершин в S без ребер, а R_T складається з вершин в T, з якого не виходить жодне з ребер). Нехай Z - сукупність вершин, досяжних в $\overrightarrow{G_y}$ з R_S. Це можна обчислити за допомогою пошуку вшир.

Якщо $R_T \cap Z$ не порожній, то значить є хоча б один шлях в $\overrightarrow{G_y}$ від R_S до R_T. Обираємо будь який з цих шляхів і так змінюємо орієнтацію всіх ребер на протилежну. Таким чином, розмір пар збільшується на 1.

Для доказів можемо додати декілька очевидних фактів:

— на обраному шляху ребра з S в T чередуються з ребрами з T в S . Це випливає з двудольності графу;

— перша вершина на шляху належить S, а остання — T. Відповідно можна допустити, що перше та останнє його ребра направлені з S в T.

За визначенням $\overrightarrow{G_y}$, це означає, що на обраному шляху ребра, що належать і не належать M чергуються, причому перше і останнє ребра не належать M ,тобто шлях являється підвищувальним, звідки і слідує твердження.

Якщо $R_T \cap Z$ порожнє, покладемо $\Delta := min\{c(i,j) - y(i) - y(j) : i \in Z \cap S, j \in T \backslash Z\}$. Δ позитивна, тому що немає жорстких ребер між $Z \cap S$ и T \ S.

Справді, для будь-якого ребра (i, j), $i \in S, j \in T$:

— якщо $i \notin Z$, $j \notin Z$, то c (i, j) -y (i) -y (j) не змінюється, бо не змінюються ні y (i), ні y (j);

— якщо $i \in Z, j \in Z$, то c (i, j) -у (i) -у (j) не змінюється, тому що у (i) збільшується на Δ, а у (j) на стільки ж зменшується;

— якщо $i \notin Z, j \in Z$ різниця c (i, j) -у (i) -у (j) збільшується на Δ, отже, залишається невід'ємною;

— якщо $i \in Z, j \notin Z$, різниця c (i, j) -у (i) -у (j) зменшується на Δ, але все одно залишається невід'ємною, тому що Δ - найменша з таких різниць.

Граф $\overrightarrow{G_y}$ змінюється, але, незважаючи на це, містить M.

Справді, щоб виключити з $\overrightarrow{G_y}$ якесь ребро (i, j), $i \in S$, $j \in T$, треба зробити його нежорсткими, тобто підвищити різницю c (i, j) -у (i) -у (j). Як ми бачили, різниця підвищується тільки якщо $i \notin Z, j \in Z$, іншими словами, і недосяжна з R_s, а j досяжна. Але в такому випадку ребро (j, i) не може належати $\overrightarrow{G_y}$, отже, ребро не належить M.

Орієнтуємо нові ребра від S до T. За визначенням Δ, безліч Z вершин, досяжних з R_s, збільшиться (при цьому число жорстких ребер зовсім не обов'язково зросте).

Для доказу цього твердження спочатку доведемо, що жодна вершина не пропаде з Z. Нехай $V \in Z$. Тоді існує шлях з якоїсь вершини, що належить R_s , в V. Всі вершини на цьому шляху досяжні зR_s, тобто належать Z. Кожне ребро на цьому шляху інцидентне двом вершинам з Z. Як ми бачили вище, для таких ребер різниця c (i, j) -у (i) -у (j) не змінюється. Значить, всі ребра шляху залишаться жорсткими, і V як і раніше буде досяжна з R_s. Тепер доведемо, що хоча б одна вершина додасться до Z. Розглянемо ребро, на якому досягається мінімум $min\{c(i,j) - y(i) - y(j): i \in Z \cap S, j \in T \backslash Z\}$.Для цього ребра, різниця c (i, j) -у (i) -у (j) обнулиться, отже, воно стане жорстким і буде направлено з S в T, тобто від i до j. Оскільки $i \in Z$, j також стане можливим з R_s, тобто додасться до Z.

Повторюємо ці кроки до тих пір, поки M не стане ідеальною парою; в цьому випадку воно дає результат з найменшою вартістю. Час виконання цієї версії алгоритму дорівнює $O(n^4)$: M доповнюється n разів, а в стадії, коли M

не змінюється, може бути не більше n змін потенціалу (так як Z збільшується кожного разу). Час, необхідний на зміну потенціалу, так само $O(n^2)$.

2.3 Математична постановка задачі

Необхідно визначити мінімально можливу вартість затребуваних ресурсів для максимально ефективного розподілення працівників між поставленими задачами та підзадачами в залежності від їх можливостей та умінь.

2.3.1 Математичний метод знаходження найефективнішого розподілення задач

Подібний розбір задачі розподілення обов'язків між працівниками з можливістю додавання визначених опцій для розподілення передбачає моделювання системи у вигляді графу(з відображенням у вигляді матриці для простоти сприйняття) та детальний розбір даних з метою знаходження оптимального рішення, яке влаштовувало б всіх – керівників і працівників.

Як вже говорилося, в даній задачі розглядається можливість додавання працівникам додаткових кваліфікацій в тій чи іншій області(збільшення кількості критеріїв), тому щоб правильно оцінити інформацію потрібно розглянути дводольні графи.

Дводольним графом (також біграфом, двочастковим графом) у математиці називається граф, множина вершин якого може бути розбита на дві підмножини так, що кожне ребро графа має одну вершину з першої підмножини і одну з другої.

Математична постановка задачі буде виглядати наступним чином.

Для n працівників та виконання робіт, дана матриця n × n, що задає вартість виконання кожної роботи кожним працівником. Знайти мінімальну вартість виконання робіт, таку що кожен працівник виконує рівно одну задачу, а кожну задачу виконує рівно один працівник.

Надалі ми під призначенням розуміємо відповідність між працівниками і завданнями, що має нульову вартість, після того як ми зробили трансформації, що впливають лише на загальну вартість робіт.

Перш за все запишемо задачу в матричній формі (Таблиця 2.1):

Таблиця 2.1 – Запис в матричній формі

a1	a2	a3	a4
b1	b2	b3	b4
c1	c2	c3	c4
d1	d2	d3	d4

де a, b, c, d - працівники, які повинні виконати роботи 1, 2, 3, 4. Коефіцієнти a1, a2, a3, a4 позначають вартість виконання працівником “a” робіт 1, 2, 3, 4 відповідно. Аналогічний сенс мають інші символи. Матриця квадратна, тому кожен працівник може виконати тільки одну задачу.

Крок 1. Зменшуємо елементи порядково. Знаходимо найменший з елементів першого рядка (a1, a2, a3, a4), і віднімаємо його з усіх елементів першого рядка. При цьому хоча б один з елементів першого рядка обнулиться. Те ж саме виконуємо і для всіх інших рядків. Тепер в кожному рядку матриці є хоча б один нуль. Іноді нулів вже досить, щоб знайти призначення. Приклад показаний в Таблиці 2.2. Нулі позначають призначені роботи.

Таблиця 2.2 – Розрахунки кроку 1

0	a2'	0	a4'
b1'	b2'	b3'	0
0	c2'	c3'	c4'
d1'	0	d3'	d4'

Крок 2. Часто на першому кроці немає призначення, як, наприклад, в наступному випадку(Таблиця 2.3):

Таблиця 2.3 – Розрахунки кроку 2

0	a2'	a3'	a4'
b1'	b2'	b3'	0
0	c2'	c3'	c4'
d1'	0	d3'	d4'

Завдання 1 може бути ефективно (за нульову вартість) виконана як працівником a, так і працівником c, зате завдання 3 не може бути ефективно виконане ніким. У таких випадках ми повторюємо крок 1 для стовпців і знову перевіряємо, чи можливе призначення.

Крок 3. У багатьох випадках ми досягнемо бажаного результату вже після кроку 2. Але іноді це не так, наприклад:

Таблиця 2.4 – Розрахунки кроку 3

0	a2'	a3'	a4'
b1'	b2'	b3'	0
0	c2'	c3'	c4'
d1'	0	0	d4'

Якщо працівник d виконує роботу 2, нема кому виконувати роботу 3, і навпаки. У таких випадках ми виконуємо процедуру, описану нижче. Спочатку, використовуючи будь-який алгоритм пошуку максимального паросполучення в дводольному графі, призначаємо якомога більше робіт тим працівникам, які можуть їх виконати за нульову вартість. Приклад показаний в Таблиці 2.5, призначені роботи виділені жирним.

Таблиця 2.5 – Друга ітерація кроку 3

0	a2'	a3'	a4'
b1'	b2'	b3'	**0**
0	c2'	c3'	c4'
d1'	**0**	0	d4'

Відзначимо всі рядки без призначень (рядок 1). Відзначимо всі стовпці з нулями в цих рядках (стовпець 1). Відзначимо всі рядки з «жирними»

нулями в цих стовпцях (рядок 3). Продовжуємо, поки нові рядки і стовпці не перестали відзначатися(Таблиця 2.6).

Таблиця 2.6 – Третя ітерація кроку 3

x				
0	a2'	a3'	a4'	x
b1'	b2'	b3'	**0**	
0	c2'	c3'	c4'	x
d1'	**0**	0	d4'	

Тепер проводимо лінії через всі відмічені стовпці і невідмічені рядки.

Таблиця 2.7 – Четверта ітерація кроку 3

x				
0	a2'	a3'	a4'	x
b1'	b2'	b3'	**0**	
0	c2'	c3'	c4'	x
d1'	**0**	0	d4'	

Всі ці дії були зроблені заради однієї мети: провести найменшу кількість ліній (вертикалей і горизонталей) так, щоб покрити всі нулі. Можна було скористатися будь-яким іншим методом замість описаного.

Крок 4. З непокритих лініями елементів матриці (в даному випадку це a2 ', a3', a4 ', c2', c3 ', c4') знайти найменший. Відняти його з усіх не зазначених рядків і додати до всіх перетинів зазначених рядків і стовпців (Таблиця 2.8). Так, наприклад, якщо найменший елемент з перерахованих дорівнює A2 ', ми отримаємо:

Таблиця 2.8 – Розрахунки кроку 4

x				
0	0	a3'-a2'	a4'-a2'	x
b1'+a2'	b2'	b3'	0	
0	c2'-a2'	c3'-a2'	c4'-a2'	x
d1'+a2'	0	0	d4'	

Це і являється основною послідовністю алгоритму для знаходження призначення задач робітникам. Оскільки вирахування призначення може бути довгим , тому потрібно повторювати процедуру (кроки 1-4) до тих пір, поки призначення не стане можливим.

2.4 Висновки до розділу

Запропоновано метод розподілення задач між працівниками враховуючи додаткові критерії у вигляді досвіду та навичок, який випливає з удосконаленого алгоритму Multilevel Feedback Queue.

Застосувавши цей алгоритм допоможе розподілити поставлені задачі в залежності від їх складності та інших опцій між правильними працівниками в майже будь якому випадку за короткий проміжок часу, що дозволяє з впевненістю сказати – цей алгоритм абсолютно підходящий для вирішення поставленого завдання.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПОДІЛЕННЯ ЗАДАЧ МІЖ РОБІТНИКАМИ

3.1 Вимоги до програмного забезпечення

Розглянуті у розділі 1 додатки, можуть вирішувати проблему розподілення задач між робітниками. При аналізі результатів можна винести деякі основні недоліки, а саме розподілення задач в напівавтоматичному режимі та перевантаження платним функціоналом.

Щоб відобразити роботи даного застосунку обрано декілька критеріїв – зацікавленість робітника в типі задачі, відповідні навички та досвід виконання задач подібного типу. Задачі будуть формуватися вручну та розподілятися за допомогою алгоритму.

При розробці додатку враховувалася можливість запускати систему на декількох платформах – Android, IOS та в подальшому – WEB – застосунки.

Після запропонованих в розділі 1 методів вирішення поставленого завдання для розподілення задач між робітниками було сформовано функціональні і нефункціональні вимоги до кінцевого результату програмного продукту. Зокрема, до нефункціональних вимог увійшли:

— закінчений програмний продукт являється мобільним додатком;

— закінчений програмний продукт містить в своїй основі модифікований алгоритм Multilevel Feedback Queue.

До функціональних вимог можна віднести:

— можливість додавання допоміжних параметрів до задач;

— можливість додавання необмеженої кількості працівників і їх параметрів;

— можливість вибору кількості задач на кожного з працівників.

На Рисунку 3.1 показана діаграма використання, яка в точності відповідає функціональним вимогам:

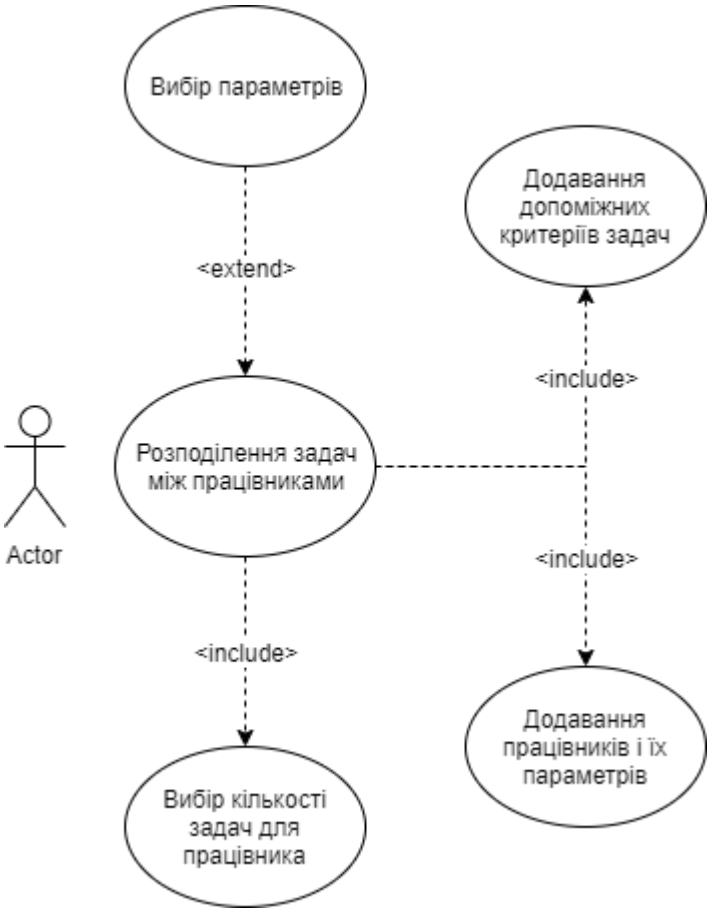


Рисунок 3.1 – Діаграма використання

Для використання великої кількості було обрано авторизацію через Gmail.

Для зберігання інформації про задачі та робітників, а також синхронізації між пристроями було обрано сервіс Google Firebase. Дані, внесені користувачами будуть збережені окремо один від одного і при необхідності будуть отримані користувачем на іншому пристрої. Наприклад, дані про завдання будуть зберігатися в структурі показаній на Рисунку 3.2.

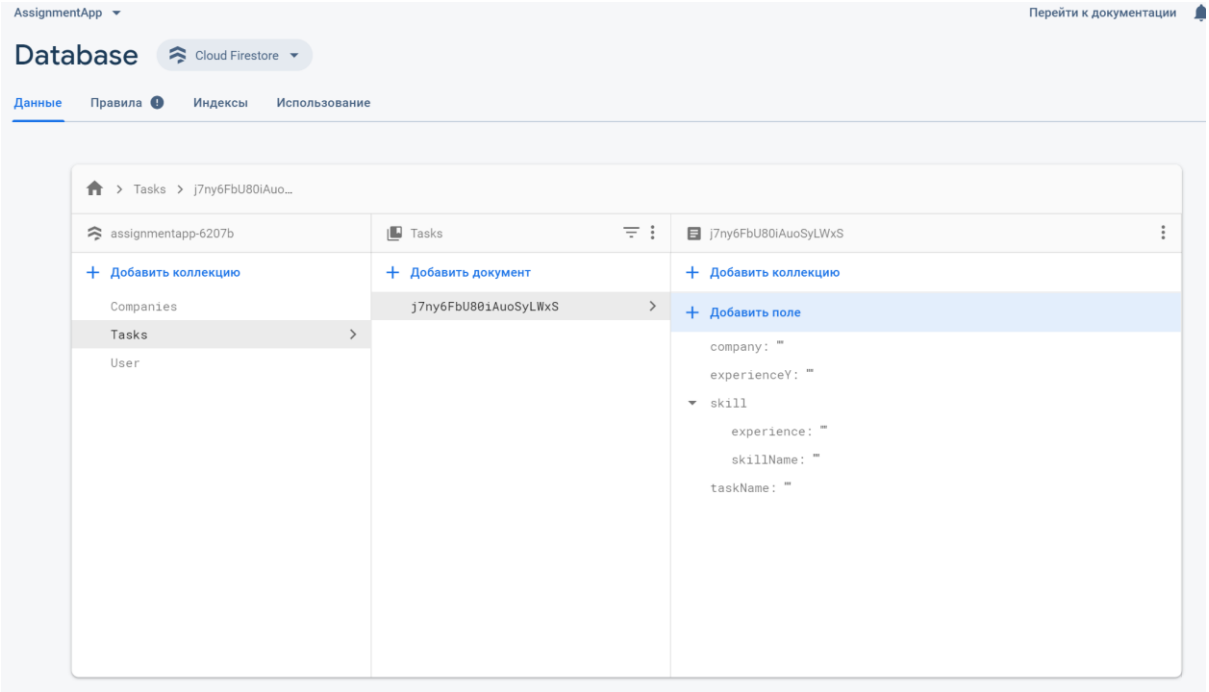


Рисунок 3.2 – Дані про завдання

Для того щоб отримати або додати необхідну нам інформацію, потрібно зробити HTTP запит типу POST на адресу https://assignmentapp-6207b.firebaseio.com/, та додати як параметри запиту поля, відповідні до зазначених в структурі.

Firebase - це платформа для розробки мобільних та веб-додатків, розроблена Firebase, Inc. у 2011 році, потім придбана Google у 2014 році. Станом на жовтень 2018 року платформа Firebase налічує 18 продуктів, які використовуються 1,5 мільйона додатків. Firebase розвивалася з Envolve, попереднього стартапу, заснованого Джеймсом Таммпліном та Ендрю Лі в 2011 році. Envolve надав розробникам API, який дозволяє інтегрувати функцію онлайн-чату на свої веб-сайти. Відмовившись від функції чату, Тамплін та Лі виявили, що його використовують для передачі даних додатків, які не були повідомленнями в чаті. Розробники використовували Envolve для синхронізації даних додатків, таких як стан гри в режимі реального часу, серед своїх користувачів. Тамплін та Лі вирішили розділити систему чатів та архітектуру в режимі реального часу, яка живила її. Вони заснували Firebase як окрему компанію у вересні 2011 року, і вона була запущена для широкого використання у квітні 2012 року.

Першим продуктом Firebase була база даних у режимі реального часу Firebase, API, який синхронізує дані додатків через iOS, Android та веб-пристрої та зберігає їх у хмарі Firebase. Продукт допомагає розробникам програмного забезпечення створювати спільні програми в режимі реального часу.

3.2 Опис архітектури програмного додатку

Додаток для мобільних пристроїв був розроблений на основі принципу клієнт – серверної архітектури.

Архітектура клієнта - сервера - це обчислювальна модель, в якій сервер розміщує, доставляє та керує більшістю ресурсів і послуг, які споживає клієнт. Цей тип архітектури має один або більше клієнтських комп'ютерів, підключених до центрального сервера через мережеве або інтернет-з'єднання. Ця система розділяє обчислювальні ресурси.

Архітектура клієнта - сервера також відома як мережева обчислювальна модель або мережа клієнт - сервер, оскільки всі запити та послуги постачаються по мережі.

В нашому варіанті розроблене програмне забезпечення складається з 2 частин:

— бібліотека з базовою архітектурою для клієнтського додатку;

— клієнтський додаток.

Опираючись на Рисунку 3.3 ми можемо бачити зв'язок компонентів, які описані вище, для програмного продукту, який розробляється.

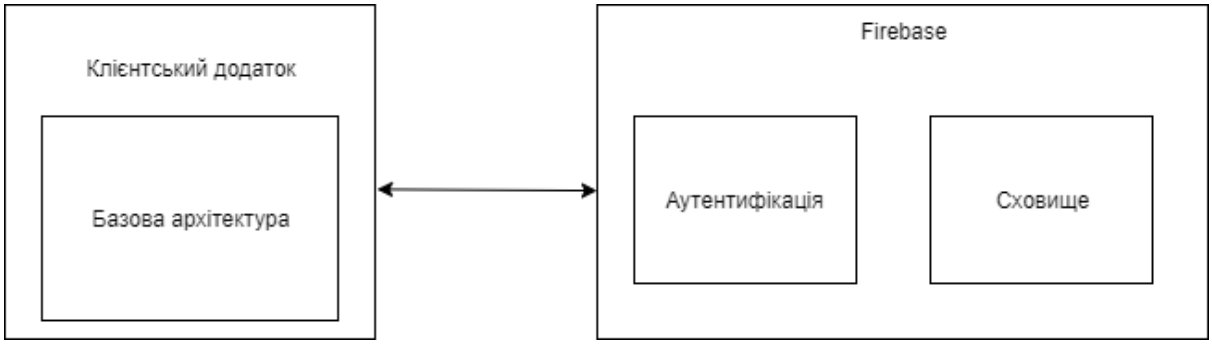


Рисунок 3.3 – Схема взаємодії між компонентами системи

Частина на стороні клієнта написана на мові програмування Kotlin з додатковою архітектурною бібліотекою на основі патерну MVV.

Kotlin - це відносно молода мова програмування від російської компанії JetBrains. З'явився він у 2011 році. На конференції Google I / O 2017 команда розробників Android повідомила, що Kotlin отримав офіційну підтримку для розробки Android-додатків.

Ось основні можливості та переваги Kotlin:

— компілюється в байткод JVM або в JavaScript;

— програми можуть використовувати всі існуючі Java-фреймворки і бібліотеки. Kotlin можна інтегрувати з Maven, Gradle і іншими системами збірки;

— мова дуже проста для вивчення;

— вихідний код відкритий;

— в IntelliJ доступна автоматична конвертація Java-коду в Kotlin і навпаки;

— мова null-безпечна - докучливі NullPointerException залишилися в Java.

— легко читаючийся синтаксис не складе проблем при code review.

Патерн MVVM (Model-View-ViewModel) дозволяє відокремити логіку додатке від візуальної частини (подання). Даний патерн є архітектурним, тобто він задає загальну архітектуру програми.

Даний патерн був представлений Джоном Госсманом в 2005 році як модифікація шаблону Presentation Model і був спочатку націлений на розробку додатків в Word Press Form(WPF). І хоча зараз цей патерн вийшов за межі WPF і застосовується в самих різних технологіях, в тому числі при розробці під Android, iOS, проте WPF є досить показовою технологією, яка розкриває можливості даного патерну.

На Рисунку 3.4 ми можемо побачити архітектуру MVVM:

Рисунок 3.4 – Архітектура MVVM

Враховуючи всі аспекти розробки даного додатку було прийнято рішення використати такі модулі, як:

— базова архітектура;

— створення задач та користувачів;

— профіль користувача;

— таблиця – схема для перегляду розподілених задач серед користувачів.

Мобільний додаток розроблювався в даному випадку під мобільну платформу Android з версією 5.0 та вище.

Android - це мобільна операційна система (ОС), вперше розроблена компанією в «Силіконовій долині» компанією під назвою Android Inc. Співпраця, очолювана Google у 2007 році через «Open Handset Alliance» (OHA), дала перевагу Android в забезпеченні повного набору програмних додатків, яке включає основну ОС, проміжне програмне забезпечення та певний мобільний застосунок.

Сама система Android дотримується концпеції відкритості для всіх – кожен користувач, який має систему на ОС Android має можливість безкоштовно розробляти програмне забезпечення та використовувати найновіші технології, які додаються в цю ОС компанією Google. В смартфонах на цій платформі присутні такі цікаві нововведення як сканер відбитку пальцю під склом екрану, розблокування телефону за допомогою інфрачервоного відбитку обличчя, камера для зйомки широкоформатної

фотографії та ін. Ця платформа охоплює близько 70% всіх сучасних мобільних девайсів, які використовуються на даний момент.

Застосунок був розроблений на основі об’єктно – орієнтованої мови програмування Kotlin. Цілісний опис функцій з базовою документацією класів міститься у Таблиці 3.1

Таблиця 3.1 – Опис базових класів додатку

Клас	**Опис**
BaseActivity	Базовий клас який описує основний функціонал активності
BaseBottomSheetDialogFragment	Клас, який описує базовий функціонал навігаційного меню
BaseDialogFragment	Клас, який описує базовий діалог в додатку
BaseFragment	Клас, що описує фрагмент для прив’язки до активності
BaseRepository	Клас, що відповідає за базовий функціонал зберігання даних
BaseViewModel	Клас, який несе відповідальність за базовий функціонал ViewModel
BindingAdapters	Клас, який відповідає за кастомні теги в xml
DICommon	Базовий клас, який описує функціонал опорного класу Dependency Injection
NetworkConnection	Клас, що відповідає за базове налаштування http запитів
FragmentViewLifecycleObserver	Клас – помічник, який допомагає отримувати актуальну інформацію при її оновленні в Fragment

Продовження таблиці 3.1

ObservableFields	Клас, який містить скорочення для найбільш використовуваних типів
Result	Клас, який містить в собі методи для роботи по отриманню результатів запиту
CommonApp	Головний клас – точка входу в додаток, містить в собі налаштування для DI та статистики
ApiConfig	Клас для зберігання налаштувань для запитів – базових URL адрес, ключів та іншого
DI	Клас, який являється тримачем всіх DI залежностей в даному додатку
AuthManager	Клас, який являється відповідальним за автентифікацію та реєстрацію
MainViewModel	Клас, який є ViewModel для MainActivity
SplashActivity	Клас, який є першим при відображенні контенту і викликається при запуску додатку
SplashViewModel	Клас, що є ViewModel для SplashActivity. Являється тримачем всієї логіки при запуску додатку

Таблиця 3.2 – Опис методів класів

Клас	**Назва**	**Вхідні параметри**	**Опис**
BaseViewModel	clearAllRx		Видаляє зв’язки з підписками
BaseViewModel	untilDestroy	handler: CompositeDisposable	Додає підписку до вже існуючої черги наглядачів
BindingAdapters	bindIsGone	view: View, isGone: Boolean	Метод для простого показу та приховування елементів
BindingAdapters	bindIsEnabled	view: View, isEnabled: Boolean	Метод для доступу на натиск елементу
DICommon	init	app: CommonApp, diHolder: IDIHolder	Ініціює модулі Dependency Injection
NetworkConnection	getNetworkSpeed		Перевіряє швидкість інтернету
NetworkConnection	getCellularConnectionSpeed	type: Int	Перевіряє швидкість інтернету в залежності від типу
NetworkConnection	getNetworkInfo	context: Context	Перевіряє підключення до інтернету
CommonApp	provideDi		Ініціює підключення DI модулів до додатку

Продовження таблиці 3.2

DI	provideApiConfig		Додає залежність ApiConfig в проект
AuthManager	login	newToken: Token	Авторизує користувача
AuthManager	refreshToken	newToken: Token	Оновляє токен авторизації
AuthManager	logout	notify: Boolean	Деавторизує користувача
MainViewModel	provideContentData		Підготовує данні для сортування
MainViewModel	makeMultilevelSort	isFullModel: Boolean	Виконує розподіл завдань для працівників
SplashActivity	makeSplashAnimation		Створює анімацію для Splash Screen

3.3 Функціонал додатку

Додаток, який ми зараз розглядаємо, представляє в собі трьохсторінковий застосунок. На кожній з сторінок розташований екран з функціоналом додатку:

— екран сортування завдань між робітниками;

— екран списку робітників;

— екран списку завдань;

— основні можливості додатку включають в себе:

— огляд розподілених задач між робітниками;

— можливість додавання критеріїв розподілення у вигляді скілів;

— можливість перерозподілу завдань та додавання скілів до вже існуючих;

Оглядаючи Рисунки 3.5-3.7 ми можемо побачити екрани застосунку з використанням функціоналу:

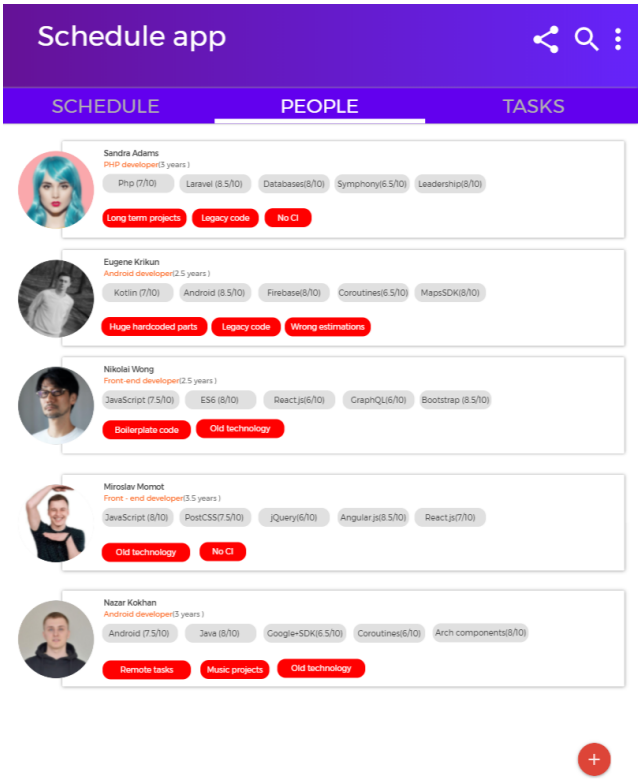


Рисунок 3.5 – Вигляд форми списку людей

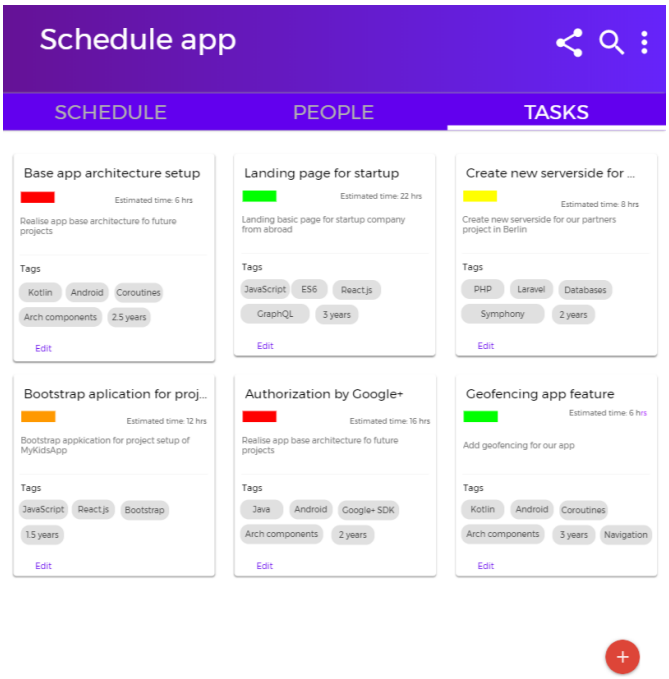


Рисунок 3.6 – Вигляд списку завдань

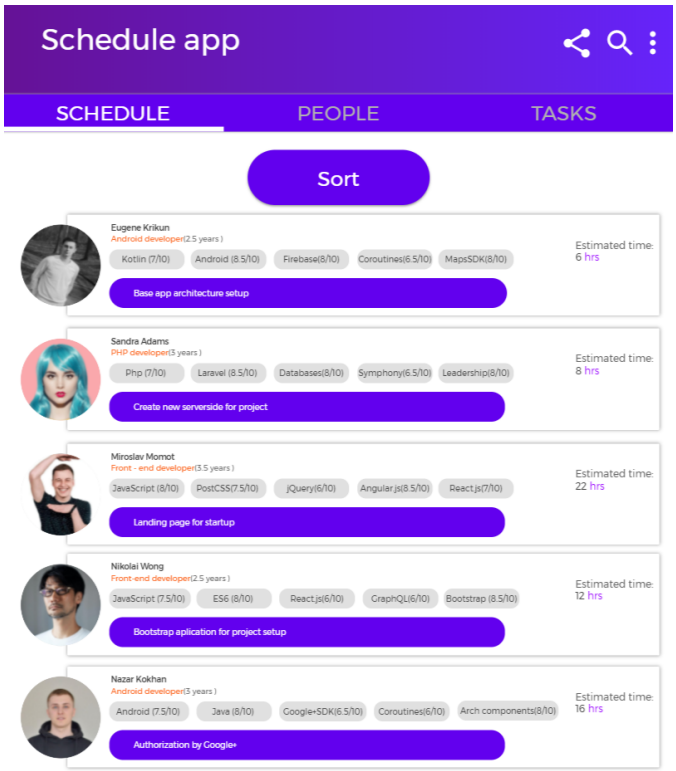


Рисунок 3.7 – Вигляд розподілених завдань між робітниками

Враховуючи те, що додаток знаходиться в розробці, то всі дії користувача записуються у вигляді логу в Firebase.

3.4 Висновки до розділу

В цьому розділі дипломної дисертації було описано загальний вигляд та функціонал реалізованого програмного продукту за вимогами які були описані, а саме: програмний продукт має бути мобільним застосунком, який має включати в себе функцію розподілення задач між робітниками з використанням методу за алгоритмом який був описаний у розділі під номером 2 цієї роботи. Ще цей додаток надає можливість додавати критерії до розподілення та пересортовувати вже готове рішення, якщо щось раптом змінилося.

Оскільки додаток включає в собі базову логіку та серверне з’єднання, а основна система являє собою сервер для обрахунків на основі Google Firebase, то для подальшої розробки можна розширити сферу використання на ще одну мобільну платформу – IOS.

Одним з функцій додатку є точна інформація про використання його користувачем шляхом логування всіх його дій.

4 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

4.1 Мета та порядок досліджень

Для того щоб випробувати розроблений програмний продукт та здійснити порівняння між результатами роботи в існуючих системах розподілення завдань, було вирішено провести низку досліджень.

Користувач додатку повинен додати завдання та робітників. Після цих дій при натисканні на кнопку “Sort” система має розподілити завдання між робітниками. В системах схожих на дану будуть проводитись аналогічні дії з попередньою похибкою на додавання завдань.

4.2 Дослідження розподілення завдань між робітниками

Оскільки існують схожі за своїм принципом системи, то і порівнювати ми будем їх між собою. Потрібно упевнитися, що результати роботи додатку співпадають, проте наша система діє як напівавтоматична в порівнянні з іншими.

Візьмемо для прикладу розподілення завдань в системі 10,000 ft.

Результати роботи даної системи показано на Рисунку 4.1

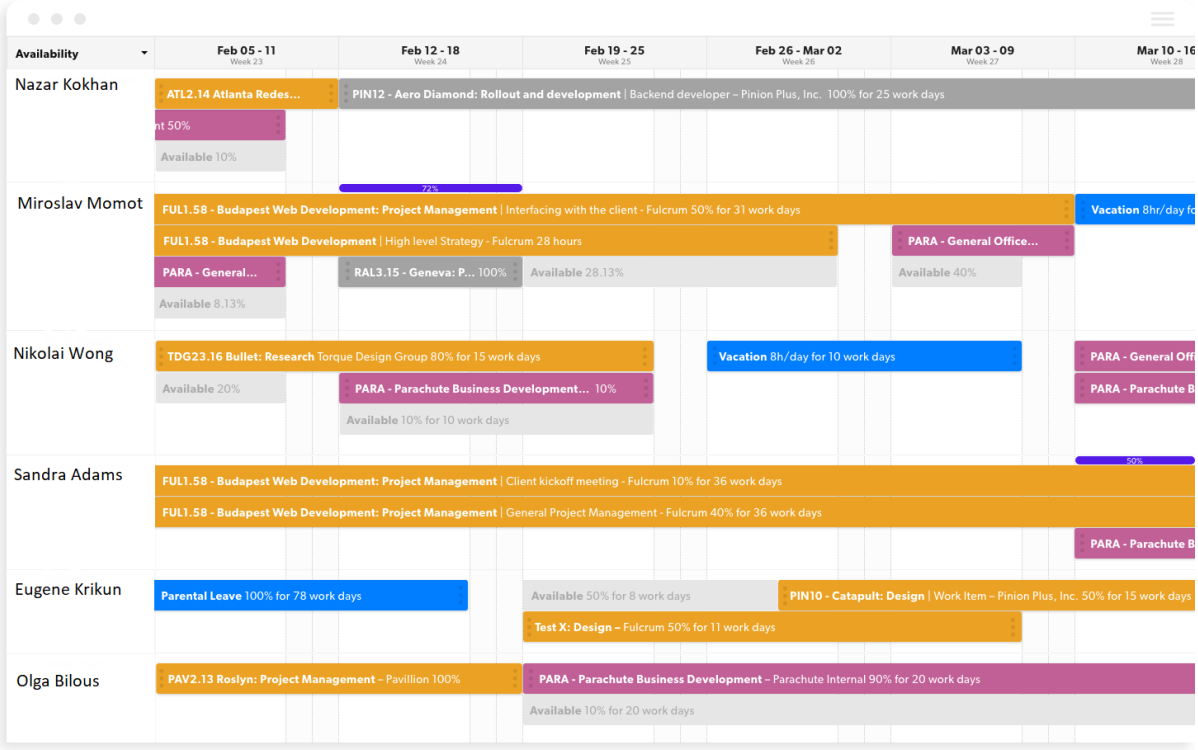


Рисунок 4.1 – Приклад роботи системи 10,000 ft

Результати роботи розробленого програмного забезпечення відображено на Рисунку 4.2

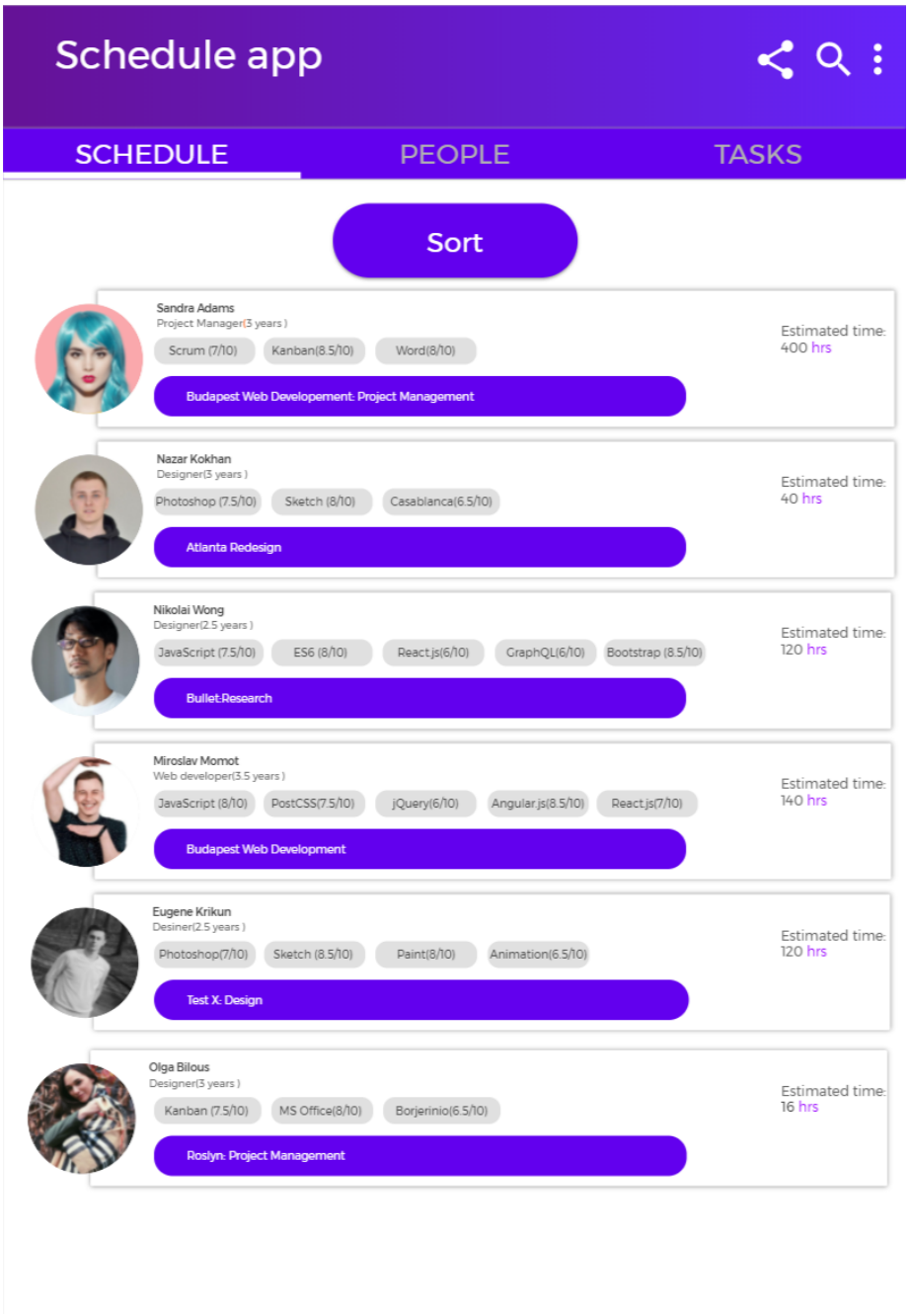


Рисунок 4.2 – Приклад роботи розробленої системи

При розподіленні завдань в системі використовувався описаний вище алгоритм, що дозволило розподілити завдання швидше за аналог у вигляді програми 10,000 ft. Також враховуючи те, що в системі 10,000 ft не

враховується досвід та навички при розподіленні являється суттєвим мінусом.

Оскільки можна сказати, що і розроблений додаток і існуючий аналог справилися з поставленим завданням, а додаткові критерії у вигляді навичок та досвіду прямопропорційно впливають на час розподілення завдань, то можна з впевненістю сказати, що системи можна порівняти за часом їх роботи і це буде справедлива оцінка для їх швидкодії як основного з критеріїв їх роботи. Дане порівняння наведено в Таблиці 4.1

Таблиця 4.1 – Огляд швидкодії систем

	Час обробки 20 завдань	Час обробки 100+ завдань	Час обробки 1000+ завдань
Розроблювана система	0,015	0,03	1,4
Saviom	0,0011	0,07	2,6
10,000 ft	0,0015	0,09	3,1
Resource Guru	0,30	1,1	4,0

Виходячи з Таблиці 4.1 можемо з впевненістю сказати, що розроблена система показала результати на 60% кращі по швидкодії, проте варто враховувати і той факт, що система враховувала досвід і вміння робітників і була швидша, тому з впевненістю можна сказати, що це успіх.

Для підтвердження отриманих результатів проведемо експеримент ще декілька разів (Таблиці 4.2 – 4.4)

Таблиця 4.2 – Повторна обробка №2

	Час обробки 20 завдань	Час обробки 100+ завдань	Час обробки 1000+ завдань
Розроблювана система	0,023	0,045	2.1
Saviom	0,0016	0,012	2,3
10,000 ft	0,002	0,016	3,2
Resource Guru	0,30	1,1	3,9

Таблиця 4.3 – Повторна обробка №3

	Час обробки 20 завдань	Час обробки 100+ завдань	Час обробки 1000+ завдань
Розроблювана система	0,2	0,67	2.8
Saviom	0,0016	0,09	2,6
10,000 ft	0,0018	0,014	3,43
Resource Guru	0,29	1,13	4,12

Таблиця 4.4 – Повторна обробка №4

	Час обробки 20 завдань	Час обробки 100+ завдань	Час обробки 1000+ завдань
Розроблювана система	0,18	0,35	2.4
Saviom	0,009	0,08	2,8
10,000 ft	0,0012	0,013	3,3
Resource Guru	0,36	1,4	4,2

4.3 Висновки до розділу

Оскільки система порівнювалась з аналогічним додатком для WEB, то можна з впевненістю сказати, що система показує кращі результати, проте можливо за кращого інтернет – з'єднання система – конкурент розробленій покаже аналогічний результат. Варто враховувати, що розроблена система використовує також спрощений варіант розподілення в якому не враховуються досвід та навички робітника.

Після огляду проведеного експерименту можна з впевненістю сказати, що додаток показує результат, який в більшості випадків кращий за аналогів. Це показує те, що він може з впевненістю конкурувати з платними конкурентами.

При порівнянні використовувалось моделювання і заповнення завдань без використання візуального інтерфейсу, тому результат дещо кращий(похибка складає +-5%).

5 РОЗРОБКА СТАРТАП –ПРОЕКТУ

5.1 Опис ідеї проекту

Сутність ідеї для даного проекту заключається в розробці та впровадженні системи автоматичного розподілення завдань між робітниками. Технологічне забезпечення відрізняється від звичайних тим, що містить в собі додатковий функціонал у вигляді розподілення завдань з додатковими критеріями шляхом автоматичного присвоєння завдання кожному з робітників.

Основними напрямками використання даної продукції є впровадження на виробництві і виробничих процесах(в основному в IT компаніях, але сфера використання необмежена).

Використана методологія розробки стартап-проекту викладена в ресурсі [1].

В Таблицях 5.1 та 5.2 наведений більш детальний опис ідеї та проведено визначення сильних, слабких та нейтральних характеристик ідеї проекту.

Таблиця 5.1 - Опис ідеї стартап-проекту

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
Розробка системи автоматичного розподілення завдань між робітниками	Впровадження на виробництві(в основному IT - компаній), також вузьконаправлених спеціалізацій, де спеціалісти являються людьми з широким спектром кваліфікацій	Зменшення вартості процесу розподілення задач. Економія часу та засобів для роботи над розподіленням задач

Таблиця 5.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	Потенційні товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Проект	Конкурент:1	Конкурент:2			
1	Ціна	+	-	-			+
2	Тривале використання	+	+	+		+	
3	Складність налагодження програмного продукту	+	-	-			+
4	Наявність замінника	+	+	+		+	

5	Час використання для отримання результату	+	-	-			+

5.2. Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (табл. 5.3):

— за якою технологією буде виготовлено товар згідно ідеї проекту?

— чи існують такі технології, чи їх потрібно розробити/доробити?

— чи доступні такі технології авторам проекту?

Таблиця 5.3 - Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових

№ п/п	**Ідея проекту**	**Технологія її реалізації**	**Наявність технології**	**Доступність технології**
1	Автоматичне розподілення завдань за допомогою модифікованого алгоритму Multilevel Feedback Queue	Розробка та впровадження автоматичного розподілення завдань за допомогою модифікованого алгоритму Multilevel Feedback Queue	Необхідно провести ретельний аналіз на визначення технології розподілення завдань з додатковими критеріями	Доступна
Обрана технологія реалізації ідеї проекту: відрізняється від традиційної				

шляхом додавання критеріїв розподілення до роботи алгоритму

На основі проведеного аналізу, можна сказати, що процес розробки такого продукту являється не дуже затратним та може бути виконаний з мінімальним забезпеченням ресурсами.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів. Спочатку проводиться аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 5.4).

Таблиця 5.4 - Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	200 - 500
2	Загальний обсяг продаж, грн./ум.од	250000-500000/рік
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Необхідність інвестицій для розробки продукту
5	Специфічні вимоги до стандартизації та сертифікації	наявні
6	Середня норма рентабельності в галузі (або по ринку), %	85%

Нижче визначені потенційні групи клієнтів, їх характеристики, та сформовано орієнтовний перелік вимог до товару для кожної групи (таблиця 5.5).

Таблиця 5.5 - Характеристика потенційних клієнтів стартап-проекту

№ п/п	**Потреба, що формує ринок**	**Цільова аудиторія**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
1	Простота використання, дешевизна, якість, швидкість роботи	Продуктові IT-компанії, орієнтовані на використання продукту для їх специфіки роботи	З огляду на потреби та фінансові можливості можна винести окремі відмінності для кожного	Надійність системи, можливість настройки під конкретного користувача, якість апробованого продукту

Після визначення груп клієнтів було проведено аналіз середовища ринку: складені таблиці з факторами, що сприяють ринковому впровадженню програмного продукту розробленого нами, та факторів, що йому перешкоджають (таблиці 5.6 - 5.7). Фактори в таблиці наведені в порядку зменшення значущості.

Таблиця 5.6 - Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Несприйняття в серйоз невеликого розробника програмного забезпечення	Постачальник програмного продукту з невеликим штатом робітників сприймається як ненадійний	Збільшення штату шляхом збільшення продажів товару
2	Відмінності технології від установлених принципів використання	Використання нової продукції викликає деякий конфуз у нових користувачів при використанні програмного продукту у порівнянні з іншим	Проведення тренінгів та воркшопів з використання продукту

Таблиця 5.7 - Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Можливість розвитку	Еволюція в самостійний і ні нащо не схожий продукт	Поліпшення системи в аспектах використання
2	Консультаційні послуги	Послуги по консультації та налагодженню продукту	Можливість притягнути додаткові кошти шляхом консультації та налагодження продукту

Нижче показано результати аналізу пропозиції: визначення загальних рис конкуренції на ринку (табл. 5.8)

Таблиця 5.8 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	**В чому проявляється дана характеристика**	**Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)**
Тип конкуренції	Чиста	Доопрацювання функціоналу продукту для переходу в монопольну або ж олігопольну конкуренцію
За рівнем конкурентоздатності	Світовий	Підприємству потрібно додати зусиль для виходу в топ найкращих підприємств
За галузевою ознакою	Внутрішньогалузева	Прикласти додаткові зусилля на визначення конкурентних можливостей, які нададуть переваги для зміцнення позицій поміж конкурентами
Конкуренція за видами товарів	Товарно - родова	Компанія веде здорову товарно – родову конкуренцію, що дозволяє їй рости і розвиватися поміж іншими підприємствами, одночасно ростячи переваги у вигляді функціоналу підчерпнутого з інших конкурентів

Продовження таблиці 5.8

За характером переваг	Цінова	Актуалізація вартості продукту
За інтенсивністю	Марочна	Ствердження компанії як бренду в майбутньому

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі.

Таблиця 5.9 - Аналіз конкуренції в галузі за М. Портером

Складові аналізу	**Прямі конкуренти в галузі**	**Потенційні конкуренти**	**Постачальники**	**Клієнти**	**Товари - замінники**
	ІТ - компанії	ІТ - компанії	ІТ - компанії	Підприємства з великою кількістю робітників	Неавтоматизовані аналоги
Висновки	Конкуренція на рівні світового ринку у вигляді ціни та функціоналу	Чистота і простота розроблюваного продукту	Удешевлення виробництва без втрати його технологічного наповнення	Використання досягнень компанії для підтримки та супроводу розробленої продукції	Підвищення якісної переваги при випуску продукції на ринок

З огляду на аналіз конкурентоздатності можна зробити висновок, що розроблена ідея має можливість роботи на ринку з огляду на конкурентну ситуацію.

Таблиця 5.10 - Обґрунтування факторів конкурентоспроможності

№ п/п/	Фактор конкурентоспроможності	Обґрунтування
1	Удешевлення вартості використання програмного продукту	Розробка, налаштування та подальший супровід програмного забезпечення надає суттєву перевагу поміж конкурентами
2	Поліпшення методики	Майбутнє поліпшення та супровід готового продукту дозволяє виділитися серед конкурентів
3	Простота технології	Користувачам легше розібратися в нашій технології, ніж в технології конкурентів

За визначеними факторами конкурентоспроможності (табл. 5.10) проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 5.11).

Таблиця 5.11 - Порівняльний аналіз сильних та слабких сторін

№ п/п/	**Фактор конкурентоспроможності**	**Бали 1-20**	**Рейтинг товарів-конкурентів у порівнянні з**						
			-3	**-2**	**-1**	**0**	**+1**	**+2**	**+3**
1	Удешевлення вартості використання програмного продукту								
2	Поліпшення методики								
3	Простота технології								

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 5.12) на

основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 5.11).

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. Наприклад: зниження доходів потенційних споживачів – фактор загрози, на основі якого можна зробити прогноз щодо посилення значущості цінового фактору при виборі товару та відповідно, – цінової конкуренції (а це вже – ринкова загроза).

Таблиця 5.12 - SWOT- аналіз стартап-проекту

Сильні сторони: Простота та вартість	**Слабкості:** Залежність від зовнішніх коштів
Можливості: Розвиток та ріст функціоналу	**Загрози:** Подавлення зі сторони більших конкурентів

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (див. табл. 5.3.6, аналіз потенційних конкурентів). Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (табл. 5.13).

Таблиця 5.13 - Альтернативи ринкового впровадження стартап-проекту

№ п/п	**Альтернатива ринкової поведінки**	**Ймовірність отримання ресурсів**	**Строки реалізації**
1	Розробка функціоналу більш швидка та якісна	Ймовірна	12 місяців
2	Підтримка функціоналу займає	Ймовірна	6 місяців

	менше часу за засобів		

Головною альтернативою після аналізу обрано розробку функціоналу більш швидко та якісно, оскільки вони включають в себе і маркетинговий і технологічний крок вперед.

5.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 5.14).

Таблиця 5.14- Вибір цільових груп потенційних споживачів

№ п/п	**Опис профілю цільової групи потенційних клієнтів**	**Готовність споживачів сприйняти продукт**	**Орієнтовний попит в межах цільової групи**	**Інтенсивність конкуренції в сегменті**	**Простота входу у сегмент**
1	ІТ - компанія	Повна	Високий	Висока	Середня
Які цільові групи обрано: ІТ - компанії					

Обрана стратегія охоплення ринку: стратегія масового маркетингу - робота з усім ринком, з розробленням для них стандартизованої програми ринкового впливу.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 5.15).

Таблиця 5.15- Визначення базової стратегії розвитку

№ п/п	**Обрана альтернатива розвитку проекту**	**Стратегія охоплення ринку**	**Ключові конкурентоспроможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку**
1	Вдосконалення функціоналу проекту	Стратегія масового маркетингу	Невпинне вдосконалення системи та програмного	Стратегія домінування в ніші

			продукту	

Наступним кроком є вибір стратегії конкурентної поведінки (таблиця 5.16)

Таблиця 5.16 - Визначення базової стратегії конкурентної поведінки

№ п/п	**Чи є проект «першопрохідцем» на ринку?**	**Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?**	**Чи буде компанія копіювати основні характеристики товару конкурента, і які?**	**Стратегія конкурентної поведінки**
1	Ні	Забирати існуючих	Так, ідею розподілення завдань	Стратегія наслідування лідерові

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див. табл. 5.5), а також в залежності від обраної базової стратегії розвитку (табл. 5.15) та стратегії конкурентної поведінки (табл. 5.16) розробляється стратегія позиціонування (табл. 5.17). що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 5.17 - Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Готовий продукт повинен повністю співпадати зі специфікацією та мати повну підтримку від виробника	Заняття ніші та можливо поступове витіснення конкурентів	Постійне вдосконалення функціоналу проекту	Доступність, простота, надійність

5.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 5.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару

Таблиця 5.18 - Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Покривати всі специфікації продукту	Доступність, надійність, підтримка	Удосконалені технології
2	Простота використання	Легкість у входженні нового користувача	На даному етапі відсутність конкуренції як такої

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 5.19).

Таблиця 5.19 - Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>Товар за задумом</td><td colspan="3"></td></tr>
<tr><td rowspan="5">Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>1 Доступність
2 Простота використання</td><td></td><td></td></tr>
<tr><td colspan="3">Якість: використання модифікованого алгоритму</td></tr>
<tr><td colspan="3">Пакування: відсутнє</td></tr>
<tr><td colspan="3">Марка: Sannacode ScheduleApp</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання:
Ліцензування виробів та цифрових копій</td></tr>
</table>

Після формування маркетингової моделі товару слід особливо відмітити чим саме проект буде захищено від копіювання. Захист може бути організовано за рахунок захисту ідеї товару (захист інтелектуальної власності), або ноу-хау, чи комплексне поєднання властивостей і характеристик, закладене на другому та третьому рівнях товару.

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (таблиця 5.20). Аналіз проводиться експертним методом.

Таблиця 5.20 - Визначення меж встановлення ціни

№ п/п	**Рівень цін на товари аналоги**	**Рівень доходів цільової групи споживачів**	**Верхня та нижня межі встановлення ціни на технологію/послугу**
1	2000$ на рік	20% економії	1000 – 1500$
2	2500$ на рік	25% економії	1500 – 2000$
3	1999$ на рік	19.5% економії	800 – 1400$

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (таблиця 5.21):

— проводити збут власними силами або залучати сторонніх;

— посередників (власна або залучена система збуту);

— вибір та обґрунтування оптимальної глибини каналу збуту;

— вибір та обґрунтування виду посередників.

Таблиця 5.21 - Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Збут своїми можливостями	Маркетингове рекламування та розповсюдження інформації про товар	міжнародний	структурована

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій, що спираються на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 5.22).

Таблиця 5.22 - Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікації, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Пошук аналогів за доступну ціну з більшим функціоналом	Сайти, розсилки, форуми	Простота, ціна, швидкість роботи	Донести до цільової аудиторії переваги даного продукту над конкурентами	Простота, швидкість, доступність

5.6 Висновки за розділом

Після розглянутого аналізу за потребами ринку, можна зробити висновок,

що даний комплекс продуктів має необхідність бути на ринку продукції, тому що він може замінити даних конкурентів за нижчу собівартість і функціонал.

Перспектива системи являється доволі високою і має право на реалізацію для міжнародного ринку.

ВИСНОВКИ

В ході роботи над магістерською дисертацією було зроблено детальний аналіз систем для розподілення задач між робітниками. Основним завданням таких програмних продуктів являється автоматичне та напівавтоматичне розподілення завдань між працівниками на підприємстві. Наявні способи реалізації є в таких системах як Saviom, 10000 ft та Resource Guru. Вони передбачають в собі додаток як невелику система для розподілення задач, а також менеджменту та редагування цих завдань. Всі ці розподілення залежать від додаткових факторів, наприклад як досвід роботи чи вміння працівника. Якщо декілька працівників мають один і той самий рівень кваліфікації для виконання поставленого завдання, то система поставить їх в чергу до більш підходящого завдання для одного з них. Величезним недоліком такого розподілення завжди були 3 критерії – час, ціна та складність використання. Ні для кого не секрет, що системи такого рівня потребуються достатнього фінансування для їх систематичного використання. Це являється основним недоліком таких систем. Також не потрібно забувати про складність і нагромадженість таких програм функціоналом, який в більшості випадків не потрібен користувачу за рідким виключенням.

Для того щоб усунути ці недоліки було проаналізовано багато методик для автоматичного розподілення завдань між робітниками.

Для розподілення використовується алгоритм Multilevel Feedback Queue з модифікацією для розподілення задач. Суть модифікації полягає в додатковому простому алгоритмі для постановки пріорітетів завданням і робітникам. Оскільки завдання розподіляються в даному випадку по навичкам та досвіду, то можна сміливо зробити припущення що розподілення з коефіцієнтною залежністю в завданнях в зв’язці алгоритмом названим вище дає суттєвий приріст в швидкості та функціонуванні алгоритму.

З використанням даного алгоритму який був промодельований було написано застосунок для мобільної платформи Android, в якому ми можемо

бачити всю зроблену роботу. Як критерії розподілення вказуються досвід та навички робітників, а також доступність його в даний момент часу до завдання.

Оскільки застосунок не являється кінцевим продуктом, то потрібно додати, що він потребує низку додаткових поправок, а саме можливості додавати як критерій час для роботи, а також систему розподілення правок до завдань. Також на майбутнє планується розширення функціоналу додатку у вигляді автоматичного розрахунку часу виконання завдання.

СПИСОК ЛІТЕРАТУРИ

1) Saviom. [Електронний ресурс] – Режим доступу: http://saviom.globalwindservice.com/

2) 10000 ft. [Електронний ресурс] – Режим доступу: https://www.10000ft.com/

3) Resource Guru ft. [Електронний ресурс] – Режим доступу: https://resourceguruapp.com/

4) Метод гілок і меж. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B3%D1%96%D0%BB%D0%BE%D0%BA_%D1%96_%D0%BC%D0%B5%D0%B6

5) Задача про розподілення. [Електронний ресурс] – Режим доступу: http://studyit2.blogspot.com/2017/01/blog-post.html

6) Задача про розподілення. [Електронний ресурс] – Режим доступу: https://ar.wikipedia.org/wiki/%D8%A7%D9%84%D8%AE%D9%88%D8%A7%D8%B1%D8%B2%D9%85%D9%8A%D8%A9_%D8%A7%D9%84%D9%85%D8%AC%D8%B1%D9%8A%D8%A9

7) Розподіл Пуассона. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D0%BF%D0%BE%D0%B4%D1%96%D0%BB_%D0%9F%D1%83%D0%B0%D1%81%D1%81%D0%BE%D0%BD%D0%B0

8) И.В. Романовский, Алгоритмы решения экстремальных задач. Вип. 7. - 2014. - С. 65-72.

9) Постановка задачи векторной оптимизации [Електронний ресурс]. – Режим доступу: http://edu.nstu.ru/courses/mo_tpr/files/5.html

10) Л. А. Гладков, Генетические алгоритмы - Вип. 1. - 2012. - С. 91-105

11) Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей. [Електронний ресурс] – Режим доступу: https://osvita.kpi.ua/files/downloads/Startup_proekt.pdf

12) Клієнт-серверна архітектура. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B097

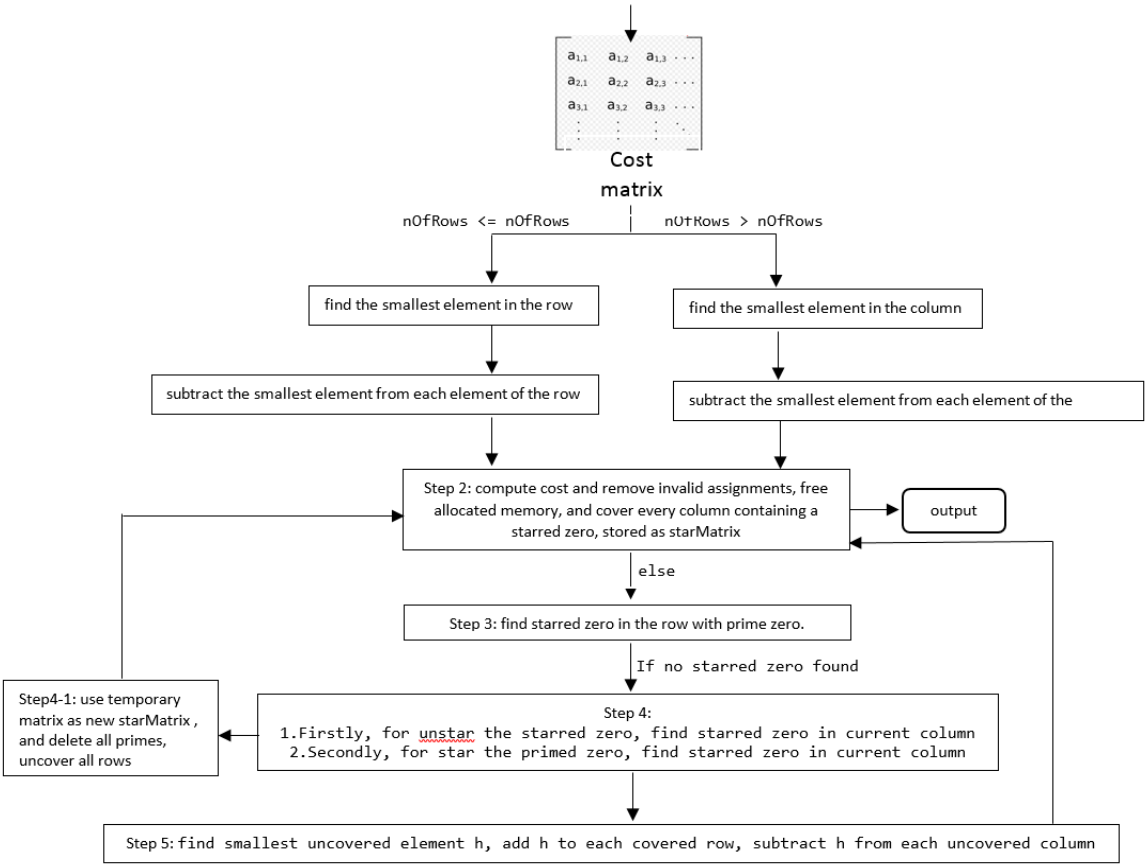
13) MVVM. [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Model-View-ViewModel

14) Android. [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Android

15) Kotlin. [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Kotlin

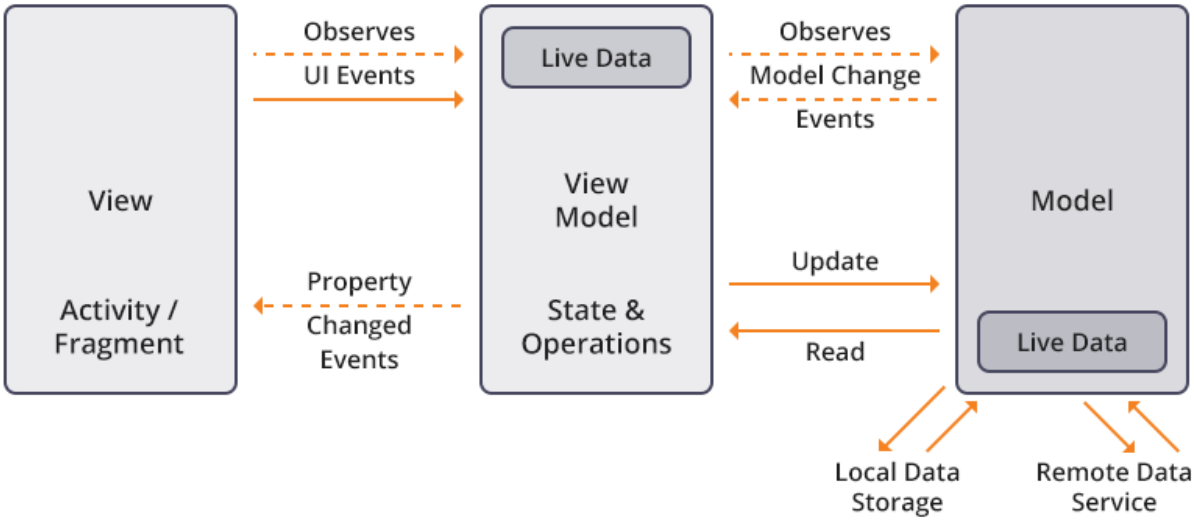
Додаток А

Алгоритм Multilevel FeedBack Queue



Додаток Б

Схема взаємодії компонентів у системі



Додаток В

Код програмного застосунку

```
import android.animation.Animator
import android.animation.ValueAnimator
import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.location.Location
import android.net.Uri
import android.os.Build
import android.os.Bundle
import android.os.Parcelable
import android.view.LayoutInflater
import android.view.View
import android.view.WindowManager
import android.view.animation.DecelerateInterpolator
import androidx.annotation.StringDef
import androidx.appcompat.app.ActionBarDrawerToggle
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.widget.Toolbar
import androidx.constraintlayout.widget.ConstraintLayout
import androidx.core.app.ShareCompat
import androidx.core.content.ContextCompat
import androidx.core.os.bundleOf
import androidx.fragment.app.Fragment
import com.akexorcist.localizationactivity.core.LocalizationActivityDelegate
import com.arellomobile.mvp.presenter.InjectPresenter
import com.crashlytics.android.answers.Answers
import com.crashlytics.android.answers.InviteEvent
import com.sannacode.android.costless.BuildConfig
import com.sannacode.android.costless.FindProductFragmentData
import com.sannacode.android.costless.R
import com.sannacode.android.costless.config.TRANSITION_SHORT
import com.sannacode.android.costless.db.DatabaseApi
import com.sannacode.android.costless.model.*
import com.sannacode.android.costless.model.response.BonusesResponse
import
com.sannacode.android.costless.model.response.LoyaltyProgramCardInfoResponse
import com.sannacode.android.costless.model.store.CategoryModel
import com.sannacode.android.costless.model.store.OrderModel
import com.sannacode.android.costless.model.store.ShopModel
import com.sannacode.android.costless.model.store.SubcategoryModel
import com.sannacode.android.costless.screen.auth.AuthActivity
import com.sannacode.android.costless.screen.common.*
import
com.sannacode.android.costless.screen.common.ImageViewerActivity.Companion.INITI
AL_POSITION
import
com.sannacode.android.costless.screen.common.ImageViewerActivity.Companion.MODE_
EXTRA
import
com.sannacode.android.costless.screen.common.ImageViewerActivity.Companion.PHOTO
S_EXTRA
import
com.sannacode.android.costless.screen.common.ImageViewerActivity.Companion.PRODU
CTS_EXTRA
import
com.sannacode.android.costless.screen.common.findproductsdialog.FindProductsFrag
ment
import
com.sannacode.android.costless.screen.common.findproductsdialog.filters.FindProd
```

```
uctFilterActivity
import
com.sannacode.android.costless.screen.common.productdetailsdialog.ProductDetails
Dialog
import
com.sannacode.android.costless.screen.common.productdetailsdialog.multishops.Pro
ductWithShopsFragment
import com.sannacode.android.costless.screen.common.pushdialog.PushDialog
import
com.sannacode.android.costless.screen.common.ratedialog.application.RateAppDialo
gFragment
import
com.sannacode.android.costless.screen.common.scanbarcode.ScanBarcodeActivity
import com.sannacode.android.costless.screen.common.webview.WebFragment
import com.sannacode.android.costless.screen.main.bonus.BonusFragment
import
com.sannacode.android.costless.screen.main.bonus.certificate.CertificateFragment
import
com.sannacode.android.costless.screen.main.bonus.certificate.select.CertificateS
electFragment
import
com.sannacode.android.costless.screen.main.bonus.history.BonusHistoryFragment
import
com.sannacode.android.costless.screen.main.bonus.history.details.BonusHistoryDet
ails
import com.sannacode.android.costless.screen.main.bonus.topup.TopUpFragment
import com.sannacode.android.costless.screen.main.cards.CardListFragment
import com.sannacode.android.costless.screen.main.cards.adding.CardAddFragment
import
com.sannacode.android.costless.screen.main.cards.common.ScanCardBarcodeActivity
import
com.sannacode.android.costless.screen.main.cards.details.CardDetailsFragment
import
com.sannacode.android.costless.screen.main.cards.details.barcodeviewer.BarcodeVi
ewerActivity
import
com.sannacode.android.costless.screen.main.cards.filters.CardFiltersFragment
import
com.sannacode.android.costless.screen.main.cards.handler.CardHandlerFragment
import
com.sannacode.android.costless.screen.main.cards.loyaltyprogram.card.LoyaltyCard
DetailsFragment
import
com.sannacode.android.costless.screen.main.cards.loyaltyprogram.congratulation.L
oyaltyCardCongratulationFragment
import
com.sannacode.android.costless.screen.main.cards.loyaltyprogram.map.LoyaltyCardM
apFragment
import
com.sannacode.android.costless.screen.main.cards.loyaltyprogram.qrcode.LoyaltyQR
CodeFragment
import
com.sannacode.android.costless.screen.main.contribution.ContributionMainFragment
import
com.sannacode.android.costless.screen.main.contribution.old.list.ContributionSto
reMainFragment
import
com.sannacode.android.costless.screen.main.contribution.product.find.Contributio
nFindProductFragment
import
com.sannacode.android.costless.screen.main.contribution.product.findbybarcode.Co
ntributionFindProductByBarcodeFragment
import
```

```
com.sannacode.android.costless.screen.main.contribution.product.handler.Contribu
tionProductHandlerFragment
import
com.sannacode.android.costless.screen.main.contribution.store.adding.Contributio
nAddStoreFragment
import com.sannacode.android.costless.screen.main.dashboard.DashboardFragment
import
com.sannacode.android.costless.screen.main.favorites.FavoriteProductsListFragmen
t
import com.sannacode.android.costless.screen.main.help.HelpFragment
import
com.sannacode.android.costless.screen.main.help.friendlyservices.FriendlyService
sFragment
import
com.sannacode.android.costless.screen.main.pricecomparison.PriceComparisonFragme
nt
import com.sannacode.android.costless.screen.main.profile.ProfileFragment
import com.sannacode.android.costless.screen.main.receipts.AddReceiptFragment
import com.sannacode.android.costless.screen.main.sales.SalesFragment
import
com.sannacode.android.costless.screen.main.sales.categoryfilters.parentcategory.
CategoryFiltersFragment
import
com.sannacode.android.costless.screen.main.sales.categoryfilters.subcategory.Sub
CategoryFiltersFragment
import
com.sannacode.android.costless.screen.main.sales.filters.SaleFiltersFragment
import
com.sannacode.android.costless.screen.main.sales.nearby.NearbySalesFragment
import com.sannacode.android.costless.screen.main.settings.SettingsFragment
import com.sannacode.android.costless.screen.main.settings.about.AboutUsFragment
import com.sannacode.android.costless.screen.main.settings.faq.FaqFragment
import
com.sannacode.android.costless.screen.main.settings.pushnotification.PushNotific
ationFragment
import
com.sannacode.android.costless.screen.main.settings.shopfilters.ShopFiltersFragm
ent
import
com.sannacode.android.costless.screen.main.settings.tutorials.TutorialsFragment
import
com.sannacode.android.costless.screen.main.shopdetails.findshop.FindShopFragment
import
com.sannacode.android.costless.screen.main.shoplists.archive.ShoppingListArchive
Fragment
import
com.sannacode.android.costless.screen.main.shoplists.details.ShoppingListDetails
Fragment
import
com.sannacode.android.costless.screen.main.shoplists.main.MainShoppingListFragme
nt
import
com.sannacode.android.costless.screen.main.shoplists.main.templates.category.Tem
plateCategoryFragment
import
com.sannacode.android.costless.screen.main.shoplists.rateshopaddreceipts.RateSho
pAddReceiptFragment
import
com.sannacode.android.costless.screen.main.shopping.common.filters.ShoppingFilte
rsFragment
import
com.sannacode.android.costless.screen.main.shopping.quick.QuickShoppingFragment
import
```

```
com.sannacode.android.costless.screen.main.shopping.quick.cart.QuickShoppingCart
Fragment
import
com.sannacode.android.costless.screen.main.shopping.simple.ShoppingFragment
import com.sannacode.android.costless.screen.main.shopreview.ShopReviewFragment
import com.sannacode.android.costless.screen.main.store.cart.CartFragment
import
com.sannacode.android.costless.screen.main.store.categories.CategoriesFragment
import
com.sannacode.android.costless.screen.main.store.categories.subcategories.Subcat
egoriesFragment
import
com.sannacode.android.costless.screen.main.store.delivery.DeliveryFragment
import
com.sannacode.android.costless.screen.main.store.orderconfirm.OrderConfirmFragme
nt
import
com.sannacode.android.costless.screen.main.store.orderfinal.OrderFinalFragment
import com.sannacode.android.costless.screen.main.store.payment.PaymentFragment
import
com.sannacode.android.costless.screen.main.store.personaldata.PersonalDataFragme
nt
import
com.sannacode.android.costless.screen.main.store.products.ProductsFragment
import com.sannacode.android.costless.screen.main.store.shops.StoreShopsFragment
import com.sannacode.android.costless.screen.main.suggestion.SuggestionFragment
import com.sannacode.android.costless.support.*
import com.sannacode.android.costless.support.AnalyticsManager.trackAction
import com.sannacode.android.costless.support.tutorial.TutorialManager
import com.sannacode.android.costless.util.*
import kotlinx.android.synthetic.main.activity_main.*
import kotlinx.android.synthetic.main.layout_dialog_message.view.*
import kotlinx.android.synthetic.main.layout_main_drawer.*
import kotlinx.android.synthetic.main.layout_main_drawer.view.*
import me.toptas.fancyshowcase.FancyShowCaseView
import org.jetbrains.anko.dip
import org.jetbrains.anko.intentFor
import ru.terrakok.cicerone.android.SupportAppNavigator
import ru.terrakok.cicerone.commands.Command
import kotlin.properties.Delegates

@StringDef(MAIN_TO_AUTH, MAIN_PROFILE, MAIN_SUGGESTION, MAIN_SETTINGS,
MAIN_SETTINGS_UNAUTHORIZED,
        MAIN_SETTINGS_ABOUT, MAIN_SETTINGS_FAQ, MAIN_SETTINGS_TERMS,
MAIN_SETTINGS_PRIVACY,
        MAIN_SETTINGS_SUGGESTION, MAIN_SALES, MAIN_SHOP_FILTERS,
MAIN_ADD_RECEIPTS, MAIN_BONUSES,
        MAIN_BONUSES_ADD_RECEIPTS, MAIN_BONUSES_HISTORY, MAIN_BONUSES_TOP_UP,
        MAIN_BONUSES_SELECT_CERTIFICATE, MAIN_BONUSES_CERTIFICATE,
MAIN_BONUSES_HISTORY_DETAILS,
        MAIN_RECEIPTS_VIEWER, MAIN_QUICK_SHOPPING, MAIN_QUICK_SHOPPING_CART,
MAIN_SHOPPING_FILTERS,
        MAIN_SHOPPING_LIST, MAIN_SHOPPING_LIST_FROM_ARCHIVE,
MAIN_SHOPPING_LIST_ARCHIVE,
        MAIN_STORE_SHOPS, MAIN_STORE_CATEGORIES, MAIN_STORE_SUBCATEGORIES,
MAIN_STORE_PRODUCTS, MAIN_STORE_CART,
        MAIN_STORE_PERSONAL_DATA, MAIN_STORE_DELIVERY, MAIN_STORE_PAYMENT,
MAIN_STORE_ORDER_CONFIRM, MAIN_STORE_ORDER_FINAL,
        MAIN_SHOPPING_LIST_DETAILS, MAIN_SHOPPING_LIST_DETAILS_FROM_GEOFENCE,
        MAIN_SHOPPING_FIND_GROCERY_STORE, MAIN_SHOPPING_FIND_GROCERY_STORE,
        MAIN_SCAN_BARCODE, MAIN_PRODUCT_VIEWER, MAIN_GO_TO_MAP, MAIN_CARDS,
MAIN_CARD_ADD_CARD,
        MAIN_CARD_HANDLER, MAIN_CARD_HANDLER_FROM_OUTSIDE,
```

```
MAIN_CARD_SCAN_BARCODE, MAIN_CARD_TAKE_PHOTO,
        MAIN_CARD_DETAILS, MAIN_CARD_VIEWER, MAIN_CARDS_FOR_SHARE,
MAIN_CARD_FILTERS,
        MAIN_SALES_FILTERS, MAIN_SALES_CATEGORY_FILTERS,
MAIN_SALES_SUBCATEGORY_FILTERS,
        MAIN_TAKE_PHOTO, MAIN_TAKE_CROPPED_PHOTO, MAIN_CONTRIBUTION_STORE_LIST,
MAIN_CONTRIBUTION_ADD_STORE, MAIN_CONTRIBUTION_ADD_PRODUCT_STORE_LIST,
        MAIN_CONTRIBUTION_FIND_PRODUCT_BY_BARCODE,
MAIN_CONTRIBUTION_FIND_PRODUCT,
        MAIN_BLOG, MAIN_SHOPPING_LIST_ADD_RECEIPTS,
MAIN_SHOPPING_LIST_FOR_SHARING, MAIN_BARCODE_VIEWER, MAIN_FILTER_ACTIVITY,
MAIN_SHOP_REVIEW, MAIN_FAVORITES, MAIN_FAVORITES_FROM_SEARCH,
MAIN_PRICE_COMPARISON,
        MAIN_SHOPPING_LIST_TEMPLATE_OPEN_DETAILS,
MAIN_SHOPPING_LIST_TEMPLATE_OPEN_CATEGORY,
MAIN_CONTRIBUTION_FROM_DETAIL_ADD_PRODUCT_STORE_LIST,
        MAIN_CONTRIBUTION_EDIT_PRODUCT_FROM_BARCODE,
MAIN_CONTRIBUTION_EDIT_PRODUCT_FROM_DETAIL,
MAIN_CONTRIBUTION_ADD_STORE_FROM_DETAIL,
        MAIN_LOYALTY_PROGRAM_CARD_DETAILS, MAIN_LOYALTY_PROGRAM_CARD_QR_CODE,
MAIN_LOYALTY_PROGRAM_CARD_CONGRATULATION, MAIN_LOYALTY_PROGRAM_CARD_MAP,
        MAIN_CONTRIBUTION_EDIT_PRODUCT_FROM_BARCODE,
MAIN_CONTRIBUTION_EDIT_PRODUCT_FROM_DETAIL,
MAIN_CONTRIBUTION_ADD_STORE_FROM_DETAIL,
        MAIN_PRODUCT_DETAILS_WITH_SHOPS, MAIN_FIND_PRODUCTS)
annotation class MainScreen

const val MAIN_TO_AUTH = "MAIN_TO_AUTH"
const val MAIN_PROFILE = "MAIN_PROFILE"
const val MAIN_PROFILE_BONUSES_HISTORY = "MAIN_PROFILE_BONUSES_HISTORY"
const val MAIN_SETTINGS = "MAIN_SETTINGS"
const val MAIN_SETTINGS_UNAUTHORIZED = "MAIN_SETTINGS_UNAUTHORIZED"
const val MAIN_SUGGESTION = "MAIN_SUGGESTION"
const val MAIN_SETTINGS_ABOUT = "MAIN_SETTINGS_ABOUT"
const val MAIN_SETTINGS_FAQ = "MAIN_SETTINGS_FAQ"
const val MAIN_SETTINGS_TERMS = "MAIN_SETTINGS_TERMS"
const val MAIN_SETTINGS_PRIVACY = "MAIN_SETTINGS_PRIVACY"
const val MAIN_SETTINGS_SUGGESTION = "MAIN_SETTINGS_SUGGESTION"
const val MAIN_SALES = "MAIN_SALES"
const val MAIN_SALES_BY_DISCOUNT_ID = "MAIN_SALES_BY_DISCOUNT_ID"
const val MAIN_SALES_BY_SHOP_NETWORK_ID = "MAIN_SALES_BY_SHOP_NETWORK_ID"
const val MAIN_SALES_NEARBY = "MAIN_SALES_NEARBY"
const val MAIN_SALES_FILTERS = "MAIN_SALES_FILTERS"
const val MAIN_SALES_FILTERS_WITH_HIDDEN_SHOP_FILTERS =
"MAIN_SALES_FILTERS_WITH_HIDDEN_SHOP_FILTERS"
const val MAIN_SALES_CATEGORY_FILTERS = "MAIN_SALES_CATEGORY_FILTERS"
const val MAIN_SALES_SUBCATEGORY_FILTERS = "MAIN_SALES_SUBCATEGORY_FILTERS"
const val MAIN_SHOP_FILTERS = "MAIN_SHOP_FILTERS"
const val MAIN_ADD_RECEIPTS = "MAIN_ADD_RECEIPTS"
const val MAIN_BONUSES = "MAIN_BONUSES"
const val MAIN_BONUSES_ADD_RECEIPTS = "MAIN_BONUSES_ADD_RECEIPTS"
const val MAIN_BONUSES_HISTORY = "MAIN_BONUSES_HISTORY"
const val MAIN_BONUSES_TOP_UP = "MAIN_BONUSES_TOP_UP"
const val MAIN_BONUSES_SELECT_CERTIFICATE = "MAIN_BONUSES_SELECT_CERTIFICATE"
const val MAIN_BONUSES_CERTIFICATE = "MAIN_BONUSES_CERTIFICATE"
const val MAIN_BONUSES_HISTORY_DETAILS = "MAIN_BONUSES_HISTORY_DETAILS"
const val MAIN_RECEIPTS_VIEWER = "MAIN_RECEIPTS_VIEWER"
const val MAIN_PHOTO_VIEWER = "MAIN_PHOTO_VIEWER"
const val MAIN_QUICK_SHOPPING = "MAIN_QUICK_SHOPPING"
const val MAIN_QUICK_SHOPPING_CART = "MAIN_QUICK_SHOPPING_CART"
const val MAIN_SHOPPING_FILTERS = "MAIN_SHOPPING_FILTERS"
const val MAIN_SHOPPING_LIST = "MAIN_SHOPPING_LIST"
const val MAIN_SHOPPING_LIST_FROM_ARCHIVE = "MAIN_SHOPPING_LIST_FROM_ARCHIVE"
```

```
const val MAIN_SHOPPING_LIST_FOR_SHARING = "MAIN_SHOPPING_LIST_FOR_SHARING"
const val MAIN_SHOPPING_LIST_ARCHIVE = "MAIN_SHOPPING_LIST_ARCHIVE"
const val MAIN_STORE_SHOPS = "MAIN_STORE_SHOPS"
const val MAIN_STORE_CATEGORIES = "MAIN_STORE_CATEGORIES"
const val MAIN_STORE_SUBCATEGORIES = "MAIN_STORE_SUBCATEGORIES"
const val MAIN_STORE_PRODUCTS = "MAIN_STORE_PRODUCTS"
const val MAIN_STORE_CART = "MAIN_STORE_CART"
const val MAIN_STORE_PERSONAL_DATA = "MAIN_STORE_PERSONAL_DATA"
const val MAIN_STORE_DELIVERY = "MAIN_STORE_DELIVERY"
const val MAIN_STORE_PAYMENT = "MAIN_STORE_PAYMENT"
const val MAIN_STORE_ORDER_CONFIRM = "MAIN_STORE_ORDER_CONFIRM"
const val MAIN_STORE_ORDER_FINAL = "MAIN_STORE_ORDER_FINAL"
const val MAIN_SHOPPING_LIST_DETAILS = "MAIN_SHOPPING_LIST_DETAILS"
const val MAIN_SHOPPING_LIST_DETAILS_FROM_GEOFENCE =
"MAIN_SHOPPING_LIST_DETAILS_FROM_GEOFENCE"
const val MAIN_SHOPPING_LIST_RATE_SHOP_ADD_RECEIPT =
"MAIN_SHOPPING_LIST_RATE_SHOP_ADD_RECEIPT"
const val MAIN_SHOPPING_LIST_TEMPLATE_OPEN_CATEGORY =
"MAIN_SHOPPING_LIST_TEMPLATE_OPEN_CATEGORY"
const val MAIN_SHOPPING_LIST_TEMPLATE_OPEN_DETAILS =
"MAIN_SHOPPING_LIST_TEMPLATE_OPEN_DETAILS"
const val MAIN_SHOPPING_FIND_GROCERY_STORE = "MAIN_SHOPPING_FIND_GROCERY_STORE"
const val MAIN_SHOPPING_COMPARE_PRICES = "MAIN_SHOPPING_COMPARE_PRICES"
const val MAIN_SCAN_BARCODE = "MAIN_SCAN_BARCODE"
const val MAIN_PRODUCT_VIEWER = "MAIN_PRODUCT_VIEWER"
const val MAIN_GO_TO_MAP = "MAIN_GO_TO_MAP"
const val MAIN_CARDS = "MAIN_CARDS"
const val MAIN_CARD_ADD_CARD = "MAIN_CARD_ADD_CARD"
const val MAIN_CARD_SCAN_BARCODE = "MAIN_CARD_SCAN_BARCODE"
const val MAIN_CARD_HANDLER = "MAIN_CARD_HANDLER"
const val MAIN_CARD_HANDLER_FROM_OUTSIDE = "MAIN_CARD_HANDLER_FROM_OUTSIDE"
const val MAIN_CARD_TAKE_PHOTO = "MAIN_CARD_TAKE_PHOTO"
const val MAIN_CARD_DETAILS = "MAIN_CARD_DETAILS"
const val MAIN_CARD_VIEWER = "MAIN_CARD_VIEWER"
const val MAIN_CARDS_FOR_SHARE = "MAIN_CARDS_FOR_SHARE"
const val MAIN_CARD_FILTERS = "MAIN_CARD_FILTERS"
const val MAIN_CONTRIBUTION = "MAIN_CONTRIBUTION"
const val MAIN_TAKE_PHOTO = "MAIN_TAKE_PHOTO"
const val MAIN_TAKE_CROPPED_PHOTO = "MAIN_TAKE_CROPPED_PHOTO"
const val MAIN_CONTRIBUTION_STORE_LIST = "MAIN_CONTRIBUTION_STORE_LIST"
const val MAIN_CONTRIBUTION_ADD_STORE = "MAIN_CONTRIBUTION_ADD_STORE"
const val MAIN_CONTRIBUTION_ADD_STORE_FROM_DETAIL =
"MAIN_CONTRIBUTION_ADD_STORE_FROM_DETAIL"
const val MAIN_CONTRIBUTION_ADD_PRODUCT_STORE_LIST =
"MAIN_CONTRIBUTION_ADD_PRODUCT_STORE_LIST"
const val MAIN_CONTRIBUTION_FROM_DETAIL_ADD_PRODUCT_STORE_LIST =
"MAIN_CONTRIBUTION_FROM_DETAIL_ADD_PRODUCT_STORE_LIST"
const val MAIN_CONTRIBUTION_FIND_PRODUCT_BY_BARCODE =
"MAIN_CONTRIBUTION_FIND_PRODUCT_BY_BARCODE"
const val MAIN_CONTRIBUTION_FIND_PRODUCT = "MAIN_CONTRIBUTION_FIND_PRODUCT"
const val MAIN_CONTRIBUTION_CREATE_PRODUCT = "MAIN_CONTRIBUTION_CREATE_PRODUCT"
const val MAIN_CONTRIBUTION_EDIT_PRODUCT = "MAIN_CONTRIBUTION_EDIT_PRODUCT"
const val MAIN_CONTRIBUTION_EDIT_PRODUCT_FROM_BARCODE =
"MAIN_CONTRIBUTION_EDIT_PRODUCT_FROM_BARCODE"
const val MAIN_CONTRIBUTION_EDIT_PRODUCT_FROM_DETAIL =
"MAIN_CONTRIBUTION_EDIT_PRODUCT_FROM_DETAIL"
const val MAIN_BLOG = "MAIN_BLOG"
const val MAIN_SHOPPING_LIST_ADD_RECEIPTS = "MAIN_SHOPPING_LIST_ADD_RECEIPTS"
const val MAIN_DASHBOARD = "MAIN_DASHBOARD"
const val MAIN_BARCODE_VIEWER = "MAIN_BARCODE_VIEWER"
const val MAIN_FILTER_ACTIVITY = "MAIN_FILTER_ACTIVITY"
const val MAIN_SHOP_REVIEW = "MAIN_SHOP_REVIEW"
const val MAIN_PUSH_SETTINGS = "PUSH_SETTINGS"
```

```
const val MAIN_TUTORIALS_SETTINGS = "MAIN_TUTORIALS_SETTINGS"
const val MAIN_FAVORITES = "MAIN_FAVORITES"
const val MAIN_FAVORITES_FROM_SEARCH = "MAIN_FAVORITES_FROM_SEARCH"
const val MAIN_PRICE_COMPARISON = "MAIN_PRICE_COMPARISON"
const val MAIN_SHOPS = "MAIN_SHOPS"
const val MAIN_HELP = "MAIN_HELP"
const val MAIN_HELP_FRIENDLY_SERVICES = "MAIN_HELP_FRIENDLY_SERVICES"
const val MAIN_LOYALTY_PROGRAM_CARD_DETAILS =
"MAIN_LOYALTY_PROGRAM_CARD_DETAILS"
const val MAIN_LOYALTY_PROGRAM_CARD_QR_CODE =
"MAIN_LOYALTY_PROGRAM_CARD_QR_CODE"
const val MAIN_LOYALTY_PROGRAM_CARD_CONGRATULATION =
"MAIN_LOYALTY_PROGRAM_CARD_CONGRATULATION"
const val MAIN_LOYALTY_PROGRAM_CARD_MAP = "MAIN_LOYALTY_PROGRAM_CARD_MAP"
const val MAIN_PRODUCT_DETAILS_WITH_SHOPS = "MAIN_PRODUCT_DETAILS_WITH_SHOPS"
const val MAIN_FIND_PRODUCTS = "MAIN_FIND_PRODUCTS"
const val MAIN_FIND_SHOP = "MAIN_FIND_SHOP"


class MainActivity : BaseApiActivity(), IMainView, INavigationHostView {
    @InjectPresenter
    lateinit var presenter: MainPresenter
    private var listMeasurement: List<Measurement> = listOf()

    //Locale helper
    override val localizationDelegate = LocalizationActivityDelegate(this)

    private var selectedViewInDrawer: View? = null
    @MainScreen
    private lateinit var activeScreen: String
    private var authorized: Boolean by Delegates.notNull()
    //Payload
    @DeepLinkManager.PayloadType
    private var deepLinkPayloadType: Int? = null
    private var deepLinkPayload: DeepLinkManager.Payload? = null
    @NotificationManager.NotificationType
    private var notificationPayloadType: String? = null
    private var notificationPayload: NotificationManager.Payload? = null

    companion object {
        private const val ACTIVE_SCREEN_EXTRA = "ACTIVE_SCREEN_EXTRA"
        private const val AUTHORIZATION_STATUS_STATE_EXTRA =
"AUTHORIZATION_STATUS_STATE_EXTRA"
        const val BEFORE_OPENED_SCREEN_EXTRA = "BEFORE_OPENED_SCREEN_EXTRA"
        const val ALERT_TYPE_EXTRA = "ALERT_TYPE"
        const val ALERT_MESSAGE_EXTRA = "ALERT_MESSAGE_EXTRA"
        const val ALERT_IMAGE_EXTRA = "ALERT_IMAGE_EXTRA"
        const val ALERT_FROM_NOTIFICATION = 2222
        const val DIALOG_FROM_NOTIFICATION = 3333
    }

    override fun getPresenters(): List<BaseApiPresenter<out IBaseApiView>>? =
listOf(presenter)

    override fun getLayout(): Int = R.layout.activity_main

    @SuppressLint("CheckResult")
    override fun init(initialStart: Boolean) {
        //setup theme

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS)
        window.clearFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS)
        window.statusBarColor = ContextCompat.getColor(this, R.color.statusBar)
```

```
        //init drawer and action bar
        drawer.setScrimColor(ContextCompat.getColor(this,
R.color.drawerScrimColor))
        //handle contribution module use permission
        drawer?.tvContribution?.postSelf {
changeVisibility(SettingsManager.isContributionPermissionAllowed()) }
        drawer?.tvAddReceipt?.postSelf {
changeVisibility(!BuildConfig.IS_LT_BUILD) }

        authorized = AuthManager.isAuthorized()
        if (authorized) {
            //init user data in drawer and subscribe to changes
            updateUserData(SettingsManager.getUser()
                    ?: AuthManager.clearToken().run { error("User is null!") })
            presenter.subscribeToUserDataUpdate()

            clProfile.isEnabled = true
            btnLogout.visible()
            btnInvite.visible()
            gProfileAuthorized.visible()
            ivLogo.gone()
            btnSignUp.gone()

            btnLogout.onClick(400) {
                trackAction(this, ACTION_CLICK, "Sign out")
                presenter.toggleAuthorization(authorized)
            }
            //handle invite stuff
            btnInvite?.onClick {
                trackAction(this, ACTION_CLICK, "Invite")
                errorSafety {
                    Answers.getInstance().logInvite(InviteEvent())
                    val inviteLink =
DeepLinkManager.createInviteLink(SettingsManager.getUser()?.fullName
                            ?: "")
                    ShareCompat.IntentBuilder.from(this)
                            .setType("text/plain")

.setChooserTitle(getString(R.string.text_invite_title))
                            .setText(inviteLink.getShortUrl(this,
DeepLinkManager.createDefaultInviteLinkProperties()))
                            .startChooser()
                }
            }

        } else {
            (ivLogo?.layoutParams as? ConstraintLayout.LayoutParams)?.topMargin
= getStatusBarHeight() + dip(20)

            clProfile.isEnabled = false
            btnLogout.gone()
            btnInvite.gone()
            gProfileAuthorized.gone()
            ivLogo.visible()
            btnSignUp.visible()

            btnSignUp.onClick(400) {
                trackAction(this, ACTION_CLICK, "Sign up")
                presenter.toggleAuthorization(authorized)
            }
        }

        //init navigation
```

```
        for ((view, screenName) in listOf(
                tvDashboard to MAIN_DASHBOARD,
                tvQuickShopping to MAIN_QUICK_SHOPPING,
                tvShops to MAIN_SHOPS,
                tvShoppingList to MAIN_SHOPPING_LIST,
                tvOnlineShopping to MAIN_STORE_SHOPS,
                tvSales to MAIN_SALES,
                tvFavorites to MAIN_FAVORITES,
                tvCalculator to MAIN_PRICE_COMPARISON,
                tvCostlessBonuses to MAIN_BONUSES,
                tvDiscountCards to MAIN_CARDS,
                tvAddReceipt to MAIN_ADD_RECEIPTS,
                tvContribution to MAIN_CONTRIBUTION,
                clProfile to MAIN_PROFILE,
                tvProfile to MAIN_PROFILE,
                tvSettings to if (authorized) MAIN_SETTINGS else
MAIN_SETTINGS_UNAUTHORIZED,
                tvFeedback to MAIN_SUGGESTION,
                tvBlog to MAIN_BLOG,
                tvHelp to MAIN_HELP)) {
            view.onClick {
                trackAction(this, ACTION_CLICK, screenName)
                if (activeScreen != screenName) {
                    presenter.handleNavigationItemClick(screenName)
                    activeScreen = screenName
                }
                drawer.close()
            }
        }

        handleLocalizedFlavors(ltFlavor = {
            tvOnlineShopping.gone()
            tvCostlessBonuses.gone()
            //tvContribution.gone()
            tvBlog.gone()
            tvCalculator.gone()
//              tvHelp.gone()
        })
        //

        //if initial start open quick shopping fragment
        if (initialStart) {
            initPayload(intent)
            val beforeOpenedScreen =
intent.getStringExtra(BEFORE_OPENED_SCREEN_EXTRA) ?: ""
            presenter.initMainActivity(true, notificationPayloadType,
notificationPayload, deepLinkPayloadType, deepLinkPayload, beforeOpenedScreen)
        } else if (::activeScreen.isInitialized) {
            onSelectNavigationItem(activeScreen) //restore menu item highlight
        }
        //

        DatabaseApi.makeDbRequest { measurementDao().getAll() }
                .subscribe({
                    if (it.isNotEmpty())
                        listMeasurement = it
                }, { it.print() })

        // show count of unwatched tutorials

updateUnwatchedTutorialsIndicator(TutorialManager.countUnwatchedTutorials())
        presenter.subscribeToUnwatchedTutorialsCount()
```

```
//        if (BuildConfig.VERSION_CODE >
SettingsManager.getLastSavedAppVersion()) {
//            if (SettingsManager.isReleaseNotesDialogEnabled()) {
//
ReleaseNotesDialogFragment.getNewInstance().show(supportFragmentManager,
ReleaseNotesDialogFragment.TAG)
//            }
//            SettingsManager.setLastSavedAppVersion(BuildConfig.VERSION_CODE)
//        }
    }

    private fun initPayload(intent: Intent) {
        deepLinkPayloadType =
intent.getIntExtra(DeepLinkManager.PAYLOAD_TYPE_EXTRA,
DeepLinkManager.PAYLOAD_NONE)
        deepLinkPayload =
intent.getParcelableExtra(DeepLinkManager.PAYLOAD_EXTRA) as?
DeepLinkManager.Payload
        notificationPayloadType =
intent.getStringExtra(NotificationManager.NOTIFICATION_TYPE_EXTRA)
        notificationPayload =
intent.getParcelableExtra(NotificationManager.NOTIFICATION_PAYLOAD_EXTRA) as?
NotificationManager.Payload
    }

    override fun onNewIntent(intent: Intent?) {
        intent?.let { initPayload(it) }
        presenter.initMainActivity(false, notificationPayloadType,
notificationPayload, deepLinkPayloadType, deepLinkPayload, "")
    }

    override fun getNavigator() = object : SupportAppNavigator(this,
supportFragmentManager, R.id.fragmentContainer) {

        override fun createActivityIntent(context: Context, @MainScreen
screenKey: String?, data: Any?): Intent? {
            fun Intent.handleMultiScreen() {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N &&
isInMultiWindowMode) {
                    addFlags(Intent.FLAG_ACTIVITY_MULTIPLE_TASK or
Intent.FLAG_ACTIVITY_LAUNCH_ADJACENT)
                }
            }
            hideKeyboard()
            @Suppress("UNCHECKED_CAST")
            return when (screenKey) {
                MAIN_TO_AUTH -> intentFor<AuthActivity>().apply {
                    val params = data as String?
                    if (params != null) {
                        putExtra(BEFORE_OPENED_SCREEN_EXTRA, params);
                    } else if (activeScreen != MAIN_FIND_PRODUCTS) {
                        putExtra(BEFORE_OPENED_SCREEN_EXTRA, activeScreen)
                    }
                    this@MainActivity.finish()
                }
                MAIN_RECEIPTS_VIEWER -> intentFor<ImageViewerActivity>().apply {
                    putExtra(MODE_EXTRA, RECEIPT_MODE)
                    val params = data as Pair<Int, List<Uri>>
                    putExtra(INITIAL_POSITION, params.first)
                    putParcelableArrayListExtra(PHOTOS_EXTRA,
ArrayList(params.second))
                }
                MAIN_PRODUCT_VIEWER -> intentFor<ImageViewerActivity>().apply {
```

```
                    putExtra(MODE_EXTRA, PRODUCT_MODE)
                    val params = data as Pair<Int, List<Product>>
                    putExtra(INITIAL_POSITION, params.first)
                    putParcelableArrayListExtra(PRODUCTS_EXTRA,
ArrayList(params.second))
                }
                MAIN_CARD_VIEWER -> intentFor<ImageViewerActivity>().apply {
                    putExtra(MODE_EXTRA, CARD_MODE)
                    val params = data as Pair<Int, Card>
                    putExtra(INITIAL_POSITION, params.first)
                    putExtra(ImageViewerActivity.CARD_EXTRA, params.second as
Parcelable)
                }
                MAIN_SCAN_BARCODE -> {
                    intentFor<ScanBarcodeActivity>().apply {
                        data?.let {
putExtra(ScanBarcodeActivity.PROCESS_BARCODE_IN_ACTIVITY, true) }
                    }
                }
                MAIN_CARD_SCAN_BARCODE -> {
                    val (shop, isFirstTime) = data as Pair<Shop?, Boolean>
                    intentFor<ScanCardBarcodeActivity>().apply {
                        shop?.let { putExtra(ScanCardBarcodeActivity.SHOP_EXTRA,
it as Parcelable) }
                        putExtra(ScanCardBarcodeActivity.FIRST_SCAN_EXTRA,
isFirstTime)
                    }
                }
                MAIN_FILTER_ACTIVITY -> {
                    intentFor<FindProductFilterActivity>()
                }
                MAIN_CARD_TAKE_PHOTO -> intentFor<CameraActivity>().apply {
                    putExtra(CameraActivity.LAUNCH_MODE_EXTRA,
CameraActivity.MODE_CARD)
                    (data as? Int)?.let {
putExtra(CameraActivity.CARD_SIDE_EXTRA, it) }
                }
                MAIN_TAKE_PHOTO -> intentFor<CameraActivity>().apply { (data as?
Bundle)?.let { putExtra(CameraActivity.BUNDLE_EXTRA, it) } }
                MAIN_TAKE_CROPPED_PHOTO -> {
                    intentFor<CameraActivity>().apply {
                        val (bundle, cropConfig) = data as Pair<Bundle,
CameraActivity.Companion.CameraRatioConfig>
                        putExtras(bundleOf(CameraActivity.LAUNCH_MODE_EXTRA to
CameraActivity.MODE_CROPPED,
                                CameraActivity.BUNDLE_EXTRA to bundle,
                                CameraActivity.RATIO_EXTRA to cropConfig))
                    }
                }
                MAIN_BARCODE_VIEWER -> {
                    intentFor<BarcodeViewerActivity>().apply {
                        //                        val (view, barcode) = data as
Pair<View, Barcode>
//                        val options =
ActivityOptionsCompat.makeSceneTransitionAnimation(this@MainActivity, view,
ViewCompat.getTransitionName(view))
//                        putExtras(options.toBundle())
                        val (barcode, storeName) = data as Pair<Barcode, String>
                        putExtra(BarcodeViewerActivity.BARCODE_EXTRA, barcode as
Parcelable)
                        putExtra(BarcodeViewerActivity.STORE_NAME_EXTRA,
storeName)
                    }
```

```
                }
                MAIN_PHOTO_VIEWER -> intentFor<ImageViewerActivity>().apply {
                    putExtra(MODE_EXTRA, PHOTO_MODE)
                    val params = data as Pair<Int, List<Uri>>
                    putExtra(INITIAL_POSITION, params.first)
                    putParcelableArrayListExtra(PHOTOS_EXTRA,
ArrayList(params.second))
                }
                else -> null
            }?.apply { handleMultiScreen() }
        }

        override fun createFragment(@MainScreen screenKey: String?, data: Any?):
Fragment {
            hideKeyboard()

            screenKey?.let { activeScreen = it }
            @Suppress("UNCHECKED_CAST")
            return when (screenKey) {
                //Main drawer items
                MAIN_DASHBOARD -> DashboardFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_QUICK_SHOPPING ->
QuickShoppingFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_SHOPS ->
com.sannacode.android.costless.screen.main.shopping.shops.ShopsFragment.getNewIn
stance()
                        .also { presenter.onSelectNavigationItem(screenKey) }
                MAIN_SHOPPING_LIST ->
MainShoppingListFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_SHOPPING_LIST_ARCHIVE ->
ShoppingListArchiveFragment.getNewInstance()
                MAIN_STORE_SHOPS -> StoreShopsFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_SALES -> SalesFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_SALES_BY_DISCOUNT_ID -> SalesFragment.getNewInstance(data
as? Long).also { presenter.onSelectNavigationItem(MAIN_SALES) }
                MAIN_SALES_BY_SHOP_NETWORK_ID ->
SalesFragment.getNewInstance(shopNetworkId = data as? Long)/*.also {
presenter.onSelectNavigationItem(MAIN_SALES) }*/
                MAIN_FAVORITES ->
FavoriteProductsListFragment.getNewInstance(false).also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_FAVORITES_FROM_SEARCH ->
FavoriteProductsListFragment.getNewInstance(true).also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_PRICE_COMPARISON ->
PriceComparisonFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_BONUSES -> BonusFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_CARDS -> CardListFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_ADD_RECEIPTS ->
AddReceiptFragment.getNewInstance(false).also {
presenter.onSelectNavigationItem(screenKey) }
                //MAIN_CONTRIBUTION ->
ContributionMainFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_CONTRIBUTION ->
```

```
ContributionMainFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_PROFILE -> ProfileFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_SETTINGS -> SettingsFragment.getNewInstance(true).also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_SETTINGS_UNAUTHORIZED ->
SettingsFragment.getNewInstance(false).also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_SUGGESTION -> SuggestionFragment.getNewInstance(false).also
{ presenter.onSelectNavigationItem(screenKey) }
                MAIN_BLOG ->
WebFragment.getNewInstance(getString(R.string.blog),
getString(R.string.blog_link, SettingsManager.getCurrentLanguage()),
                        false, WebFragment.Companion.Header("blog-mobile",
true)).also { presenter.onSelectNavigationItem(screenKey) }
                //Quick shopping
                MAIN_QUICK_SHOPPING_CART ->
QuickShoppingCartFragment.getNewInstance()
                MAIN_SHOPPING_FILTERS ->
ShoppingFiltersFragment.getNewInstance(data as Int)
                MAIN_SHOP_REVIEW -> ShopReviewFragment.getNewInstance(true, data
as ShopFullModel)
                //Shops
                MAIN_FIND_SHOP -> FindShopFragment.getNewInstance(data as
ShopFull)
                //Shopping List
                MAIN_SHOPPING_LIST_FROM_ARCHIVE ->
MainShoppingListFragment.getNewInstance().also {
onSelectNavigationItem(MAIN_SHOPPING_LIST) }
                MAIN_SHOPPING_LIST_FOR_SHARING ->
MainShoppingListFragment.getNewInstance(data as Long).also {
onSelectNavigationItem(MAIN_SHOPPING_LIST) }
                MAIN_SHOPPING_LIST_DETAILS ->
ShoppingListDetailsFragment.getNewInstance(data as ShoppingList, listMeasure =
listMeasurement)
                MAIN_SHOPPING_LIST_DETAILS_FROM_GEOFENCE ->
ShoppingListDetailsFragment.getNewInstance(data as ShoppingList, true,
listMeasure = listMeasurement)
                MAIN_SHOPPING_FIND_GROCERY_STORE ->
ShoppingFragment.getNewInstance(data as List<ProductFull>, shoppingListLocalID =
(data as List<ProductShoppingList>).first().shoppingListID)
                MAIN_SHOPPING_COMPARE_PRICES ->
ShoppingFragment.getNewInstance(data as List<ProductFull>, true, (data as
List<ProductShoppingList>).first().shoppingListID)
                MAIN_SHOPPING_LIST_ADD_RECEIPTS ->
RateShopAddReceiptFragment.getNewInstance(data as ShoppingListModel, true)
                MAIN_SHOPPING_LIST_RATE_SHOP_ADD_RECEIPT ->
RateShopAddReceiptFragment.getNewInstance(data as ShoppingListModel)
                //Online store
                MAIN_STORE_CATEGORIES -> CategoriesFragment.getNewInstance(data
as ShopModel)
                MAIN_STORE_SUBCATEGORIES ->
SubcategoriesFragment.getNewInstance(data as Pair<ShopModel, CategoryModel>)
                MAIN_STORE_PRODUCTS -> ProductsFragment.getNewInstance(data as
Pair<ShopModel, SubcategoryModel>)
                MAIN_STORE_CART -> CartFragment.getNewInstance(data as
Pair<ShopModel, Boolean>)
                MAIN_STORE_PERSONAL_DATA ->
PersonalDataFragment.getNewInstance(data as Pair<ShopModel, Double>)
                MAIN_STORE_DELIVERY -> DeliveryFragment.getNewInstance(data as
Pair<ShopModel, OrderModel>)
                MAIN_STORE_PAYMENT -> PaymentFragment.getNewInstance(data as
```

```
Pair<ShopModel, Int>)
                MAIN_STORE_ORDER_CONFIRM ->
OrderConfirmFragment.getNewInstance(data as Pair<ShopModel, OrderModel>)
                MAIN_STORE_ORDER_FINAL -> OrderFinalFragment.getNewInstance(data
as String)
                //Shopping List Template
                MAIN_SHOPPING_LIST_TEMPLATE_OPEN_CATEGORY ->
TemplateCategoryFragment.getNewInstance(data as CategoryTemplate)
                MAIN_SHOPPING_LIST_TEMPLATE_OPEN_DETAILS ->
ShoppingListDetailsFragment.getNewInstance(data as ShoppingList, isTemplate =
true, listMeasure = listMeasurement)
                //Bonuses
                MAIN_BONUSES_ADD_RECEIPTS ->
AddReceiptFragment.getNewInstance(true)
                MAIN_BONUSES_HISTORY -> BonusHistoryFragment.getNewInstance(data
as? Int)
                MAIN_PROFILE_BONUSES_HISTORY ->
BonusHistoryFragment.getNewInstance(null, true)
                MAIN_BONUSES_TOP_UP -> TopUpFragment.getNewInstance(data as Int)
                MAIN_BONUSES_SELECT_CERTIFICATE ->
CertificateSelectFragment.getNewInstance(data as BonusesResponse)
                MAIN_BONUSES_CERTIFICATE -> {
                    val dataFormatted = data as Pair<Int, ShopNetworkBonuses>
                    CertificateFragment.getNewInstance(dataFormatted.first,
dataFormatted.second)
                }
                MAIN_BONUSES_HISTORY_DETAILS ->
BonusHistoryDetails.getNewInstance(data as EarningModel)
                //Cards
                MAIN_CARD_ADD_CARD -> CardAddFragment.getNewInstance()
                MAIN_CARD_HANDLER -> {
                    val (card, shop, barcode) = data as Triple<Card?, Shop?,
Barcode?>
                    CardHandlerFragment.getNewInstance(card, shop, barcode)
                }
                MAIN_CARD_HANDLER_FROM_OUTSIDE -> {
                    val (shop, barcode) = data as Pair<Shop?, Barcode?>
                    CardHandlerFragment.getNewInstance(shop = shop, barcode =
barcode, isCardNativeFlow = false)
                }
                MAIN_CARD_DETAILS -> CardDetailsFragment.getNewInstance(data as
Card)
                MAIN_CARDS_FOR_SHARE -> CardListFragment.getNewInstance(data as?
String).also { presenter.onSelectNavigationItem(MAIN_CARDS) }
                MAIN_CARD_FILTERS -> CardFiltersFragment.getNewInstance()
                //Loyalty program
                MAIN_LOYALTY_PROGRAM_CARD_DETAILS ->
LoyaltyCardDetailsFragment.getNewInstance(data as LoyaltyProgramCardModel)
                MAIN_LOYALTY_PROGRAM_CARD_QR_CODE ->
LoyaltyQRCodeFragment.getNewInstance(data as Pair<LoyaltyProgramCardModel,
String>)
                MAIN_LOYALTY_PROGRAM_CARD_CONGRATULATION ->
LoyaltyCardCongratulationFragment.getNewInstance(data as String)
                MAIN_LOYALTY_PROGRAM_CARD_MAP ->
LoyaltyCardMapFragment.getNewInstance(data as LoyaltyProgramCardInfoResponse)
                //Sales
                MAIN_SALES_FILTERS ->
SaleFiltersFragment.getNewInstance(AuthManager.isAuthorized())
                MAIN_SALES_FILTERS_WITH_HIDDEN_SHOP_FILTERS ->
SaleFiltersFragment.getNewInstance(AuthManager.isAuthorized(), true)
                MAIN_SALES_CATEGORY_FILTERS ->
CategoryFiltersFragment.getNewInstance()
                MAIN_SALES_SUBCATEGORY_FILTERS ->
```

```
SubCategoryFiltersFragment.getNewInstance(data as? ParentCategorySalesModel)
                MAIN_SALES_NEARBY -> (data as? Long)?.let {
NearbySalesFragment.getNewInstance(it) }
                        ?: NearbySalesFragment.getNewInstance()
                //Contribution
                MAIN_CONTRIBUTION_STORE_LIST ->
ContributionStoreMainFragment.getNewInstance()
                MAIN_CONTRIBUTION_ADD_PRODUCT_STORE_LIST ->
ContributionMainFragment.getNewInstance()
                MAIN_CONTRIBUTION_ADD_STORE ->
ContributionAddStoreFragment.getNewInstance(data as? Location)
                MAIN_CONTRIBUTION_ADD_STORE_FROM_DETAIL ->
ContributionAddStoreFragment.getNewInstance(data as? Location, true)

                MAIN_CONTRIBUTION_FROM_DETAIL_ADD_PRODUCT_STORE_LIST -> {
                    val (isFromBarcode, product) = data as Pair<Boolean?,
ContributionProductModel?>
                    ContributionMainFragment.getNewInstance(isFromBarcode,
product, true)
                }

                MAIN_CONTRIBUTION_FIND_PRODUCT_BY_BARCODE ->
ContributionFindProductByBarcodeFragment.getNewInstance(data as
ContributionStore)
                MAIN_CONTRIBUTION_FIND_PRODUCT -> {
                    val (store, barcode) = data as Pair<ContributionStore,
Barcode?>
                    ContributionFindProductFragment.getNewInstance(store,
barcode)
                }
                MAIN_CONTRIBUTION_CREATE_PRODUCT -> {
                    val (store, product, barcode) = data as
Triple<ContributionStore, ContributionProduct, Barcode?>
                    ContributionProductHandlerFragment.getNewInstance(store,
product, ContributionProductHandlerFragment.CREATE_PRODUCT_MODE, barcode)
                }
                MAIN_CONTRIBUTION_EDIT_PRODUCT -> {
                    val (store, product, barcode) = data as
Triple<ContributionStore, ContributionProduct, Barcode?>
                    ContributionProductHandlerFragment.getNewInstance(store,
product, ContributionProductHandlerFragment.EDIT_PRODUCT_MODE, barcode)
                }
                //FROM DETAIL
                MAIN_CONTRIBUTION_EDIT_PRODUCT_FROM_BARCODE -> {
                    val (store, product, barcode) = data as
Triple<ContributionStore, ContributionProduct, Barcode?>
                    ContributionProductHandlerFragment.getNewInstance(store,
product, ContributionProductHandlerFragment.EDIT_PRODUCT_MODE, barcode, true)
                }

                MAIN_CONTRIBUTION_EDIT_PRODUCT_FROM_DETAIL -> {
                    val (store, product, barcode) = data as
Triple<ContributionStore, ContributionProduct, Barcode?>
                    ContributionProductHandlerFragment.getNewInstance(store,
product, ContributionProductHandlerFragment.EDIT_PRODUCT_MODE, barcode,
isFromBarcode = false, isFromDetail = true)
                }

                //Settings
                MAIN_SETTINGS_ABOUT -> AboutUsFragment.getNewInstance()
                MAIN_SETTINGS_FAQ -> FaqFragment.getNewInstance()
                MAIN_SETTINGS_TERMS ->
WebFragment.getNewInstance(getString(R.string.text_terms),
```

```
getString(R.string.terms_link, SettingsManager.getCurrentLanguage()))
                MAIN_SETTINGS_PRIVACY ->
WebFragment.getNewInstance(getString(R.string.privacy_title),
getString(R.string.privacy_link, SettingsManager.getCurrentLanguage()))
                MAIN_SETTINGS_SUGGESTION ->
SuggestionFragment.getNewInstance(true)
                MAIN_SHOP_FILTERS -> ShopFiltersFragment.getNewInstance()
                MAIN_PUSH_SETTINGS -> PushNotificationFragment.getNewInstance()
                MAIN_TUTORIALS_SETTINGS -> TutorialsFragment.getNewInstance()
                //Help
                MAIN_HELP -> HelpFragment.getNewInstance().also {
presenter.onSelectNavigationItem(screenKey) }
                MAIN_HELP_FRIENDLY_SERVICES ->
FriendlyServicesFragment.getNewInstance()
                MAIN_PRODUCT_DETAILS_WITH_SHOPS -> {
                    try {
                        val (product, canBeUsedInApp) = data as
Pair<ProductFull, Boolean>
                        ProductWithShopsFragment.getNewInstance(product, barcode
= null, canBeUsedInApp = canBeUsedInApp, handleFavorite = true, isFromBarcode =
false)
                    } catch (e: Exception) {
                        e.print()
                        DashboardFragment.getNewInstance()
                    }
                }
                MAIN_FIND_PRODUCTS -> {
                    FindProductsFragment.getNewInstance(data as?
FindProductFragmentData
                            ?: FindProductFragmentData(mode =
FindProductsFragment.QUICK_SHOPPING_MODE))
                }
                //
                else -> DashboardFragment.getNewInstance()
            }
        }

        override fun setupFragmentTransactionAnimation(command: Command?,
currentFragment: Fragment?, nextFragment: Fragment?, fragmentTransaction:
androidx.fragment.app.FragmentTransaction?) {

fragmentTransaction?.setTransition(androidx.fragment.app.FragmentTransaction.TRA
NSIT_FRAGMENT_FADE)
        }

    }

    override fun onSelectNavigationItem(screen: String) {
        fun selectActiveItemUi(view: View) {
            for (item in listOf(
                    tvDashboard, tvQuickShopping, tvShops, tvShoppingList,
tvOnlineShopping, tvSales, tvFavorites, tvCalculator,  // 1 section
                    tvCostlessBonuses, tvDiscountCards, tvAddReceipt,
tvContribution,                                                 // 2 section
                    tvProfile, tvSettings, tvFeedback, tvBlog, tvHelp)) {
// 3 section
                item.isActivated = false
            }
            view.isActivated = true
            selectedViewInDrawer = view
        }

        selectActiveItemUi(when (screen) {
```

```
            MAIN_DASHBOARD -> tvDashboard
            MAIN_QUICK_SHOPPING -> tvQuickShopping
            MAIN_SHOPS -> tvShops
            MAIN_SHOPPING_LIST -> tvShoppingList
            MAIN_STORE_SHOPS -> tvOnlineShopping
            MAIN_SALES -> tvSales
            MAIN_FAVORITES -> tvFavorites
            MAIN_PRICE_COMPARISON -> tvCalculator
            MAIN_BONUSES -> tvCostlessBonuses
            MAIN_CARDS -> tvDiscountCards
            MAIN_ADD_RECEIPTS -> tvAddReceipt
            MAIN_CONTRIBUTION -> tvContribution
            MAIN_PROFILE -> tvProfile
            MAIN_SETTINGS -> tvSettings
            MAIN_SETTINGS_UNAUTHORIZED -> tvSettings
            MAIN_SUGGESTION -> tvFeedback
            MAIN_BLOG -> tvBlog
            MAIN_HELP -> tvHelp
            else -> tvDashboard
        })
    }

    fun configureDrawerMenuButton(toolbar: Toolbar, isMenuAvailable: Boolean) {
        val toggle = object : ActionBarDrawerToggle(this, drawer, toolbar,
R.string.drawer_open, R.string.drawer_close) {
            var isKeyboardClose: Boolean = false

            override fun onDrawerSlide(drawerView: View, slideOffset: Float) {
                if (!isKeyboardClose && slideOffset > 0.33f) {
                    hideKeyboard()
                }
                super.onDrawerSlide(drawerView, slideOffset)
            }

            override fun onDrawerClosed(drawerView: View) {
                super.onDrawerClosed(drawerView)
                isKeyboardClose = false
            }
        }

        if (presenter.lastToggleButtonState != isMenuAvailable) {
            if (isMenuAvailable) {
                supportActionBar?.setDisplayHomeAsUpEnabled(true)
                drawer.unlock()
            } else {
                toggle.apply {
                    isDrawerIndicatorEnabled = true
                }
                toggle.syncState()
                drawer.lock()
            }

            val anim = if (isMenuAvailable) {
                ValueAnimator.ofFloat(1F, 0F)
            } else {
                ValueAnimator.ofFloat(0F, 1F)
            }
            with(anim) {
                addUpdateListener { valueAnimator ->
                    val slideOffset = valueAnimator.animatedValue as Float
                    toggle.onDrawerSlide(drawer, slideOffset)
                }
                addListener(object : Animator.AnimatorListener {
```

```
                    override fun onAnimationRepeat(p0: Animator?) {

                    }

                    override fun onAnimationCancel(p0: Animator?) {
                    }

                    override fun onAnimationStart(p0: Animator?) {
                        toggle.isDrawerSlideAnimationEnabled = true
                        if (isMenuAvailable) {
                            supportActionBar?.setDisplayHomeAsUpEnabled(false)
                            toggle.isDrawerIndicatorEnabled = true
                            toggle.syncState()
                        }
                    }

                    override fun onAnimationEnd(p0: Animator?) {
                        if (isMenuAvailable) {
                            toggle.isDrawerSlideAnimationEnabled = false
                        } else {
                            enableToggleAsBackButton(toggle)
                        }
                    }
                })
                interpolator = DecelerateInterpolator()
                duration = TRANSITION_SHORT
                start()
            }
        } else {
            if (isMenuAvailable) {
                toggle.apply {
                    isDrawerSlideAnimationEnabled = false
                    isDrawerIndicatorEnabled = true
                }
                toggle.syncState()
                drawer.unlock()
            } else {
                enableToggleAsBackButton(toggle)
            }
        }

        drawer.addDrawerListener(toggle)
        presenter.lastToggleButtonState = isMenuAvailable
    }

    private fun enableToggleAsBackButton(toggle: ActionBarDrawerToggle) {
        toggle.isDrawerIndicatorEnabled = false
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        toggle.setToolbarNavigationClickListener {
            trackAction(this@MainActivity, ACTION_CLICK, "On back pressed")
            onBackPressed()
        }
        drawer.lock()
    }

    override fun toggleContributionModule(isAllowed: Boolean) {
        errorSafety {
            drawer?.tvContribution?.postSelf { errorSafety {
changeVisibility(isAllowed) } }
        }
    }

    override fun toggleDrawer(show: Boolean) {
```

```
        if (show) {
            drawer?.open()
        } else {
            drawer?.close()
        }
    }

    override fun updateUserData(user: UserModel) {
        (flAvatar?.layoutParams as? ConstraintLayout.LayoutParams)?.topMargin =
getStatusBarHeight() + dip(20)
        ivAvatar?.loadImage(user.avatarPath.toUri(), R.drawable.avatar_vd,
R.drawable.avatar_vd, fit = false)
        tvName?.text = user.fullName
        tvEmail?.text = user.email
    }

    override fun updateUnwatchedTutorialsIndicator(count: Int) {
        if (count > 0) {
            tvTutorialsIndicator.text = count.toString()
            tvTutorialsIndicator.visible()
        } else {
            tvTutorialsIndicator.gone()
        }
    }

    override fun handleBackStack() {
        if (drawer.isOpen) {
            drawer.close()
        } else {
            super.handleBackStack()
        }
    }

    override fun showProgress() {
        getProgressView()?.visible()
    }

    override fun hideProgress() {
        getProgressView()?.gone()
    }

    override fun errorMessage(message: String) {
    }

    override fun successMessage(message: String) {
    }

    fun getProgressView(): View? = clProgressFullScreen

    override fun showAlert(show: Boolean, params: Bundle?) {
        if (show) {
            if (params != null && params.getInt(ALERT_TYPE_EXTRA) ==
ALERT_FROM_NOTIFICATION) { // Alert
                val messageView =
LayoutInflater.from(this).inflate(R.layout.layout_dialog_message, null)
                val messageText = params.getString(ALERT_MESSAGE_EXTRA)
                val image: String? = params.getString(ALERT_IMAGE_EXTRA)
                messageView.tvMessage.text = messageText
                image?.let {
                    val imageUri = it.toUri()
                    with(messageView.ivImage) {
                        loadImageWithProgress(imageUri, hideOnFail = true)
                        visible()
```

```
                        onClick { getRouter().navigateTo(MAIN_PHOTO_VIEWER, 0 to
listOf(imageUri)) }
                    }
                }

                AlertDialog.Builder(this)
                        .setView(messageView)
                        .setCancelable(true)
                        .setOnCancelListener { presenter.hideAlert() }
                        .setPositiveButton(R.string.text_ok) { dialog, _ ->
dialog.cancel() }
                        .saveShow()
            } else if (params != null && params.getInt(ALERT_TYPE_EXTRA) ==
DIALOG_FROM_NOTIFICATION) { // Dialog
                val messageText = params.getString(ALERT_MESSAGE_EXTRA)
                val image: String = params.getString(ALERT_IMAGE_EXTRA) ?:
return

                //open dialog
                PushDialog.create(messageText,
image).show(supportFragmentManager, PushDialog.TAG)
            } else {

AlertDialog.Builder(this).setMessage(getString(R.string.register_to_continue))
                        .setCancelable(true)
                        .setOnCancelListener { presenter.hideAlert() }
                        .setPositiveButton(getString(R.string.sign_up)) {
dialog, _ ->
                            dialog.cancel()
                            presenter.toggleAuthorization(authorized)
                        }

.setNegativeButton(com.sannacode.android.costless.util.getString(R.string.text_c
ancel)) { dialog, _ -> dialog.cancel() }.saveShow()
            }
        }
    }

    override fun showRateAppDialog() {
        RateAppDialogFragment.getNewInstance("").show(supportFragmentManager,
RateAppDialogFragment.TAG)
    }

    override fun handleDialogsWhenOpeningNotifications() {

        fun closeAllDialogs(fragmentManager:
androidx.fragment.app.FragmentManager) {
            fragmentManager.fragments.forEach {
                if (it is androidx.fragment.app.DialogFragment) {
                    it.dismiss()
                }
            }
        }

        supportFragmentManager?.fragments?.forEach {
            if (it is androidx.fragment.app.DialogFragment) {
                it.dismiss()
            } else {
                closeAllDialogs(it.childFragmentManager)
            }
        }
        FancyShowCaseView.hideCurrent(this)
    }
```

```
    override fun onShareSaleProcessed(saleModel: SaleStandaloneModel) {
        ProductDetailsDialog.create(saleModel).show(supportFragmentManager,
ProductDetailsDialog.TAG)
    }

//    override fun onConfigurationChanged(newConfig: Configuration?) {
//        super.onConfigurationChanged(newConfig)
//
//    }

    override fun updateMeasureFromDB() {
        DatabaseApi.makeDbRequest { measurementDao().getAll() }
                .subscribe({
                    if (!it.isEmpty())
                        listMeasurement = it
                }, { it.print() })
    }


    /*Saving data on configuration changes*/
    override fun onSaveInstanceState(outState: Bundle?) {
        super.onSaveInstanceState(outState)
        outState?.apply {
            if (::activeScreen.isInitialized) {
                putString(ACTIVE_SCREEN_EXTRA, activeScreen)
            }

            putBoolean(AUTHORIZATION_STATUS_STATE_EXTRA, authorized)
        }

    }

    override fun onRestoreInstanceState(savedInstanceState: Bundle?) {
        super.onRestoreInstanceState(savedInstanceState)
        savedInstanceState?.apply {
            activeScreen = getString(ACTIVE_SCREEN_EXTRA)
            authorized = getBoolean(AUTHORIZATION_STATUS_STATE_EXTRA)
        }
    }
}
```